#153A

E7.4-10627

CR-138737

ERTS IMAGE DATA COMPRESSION TECHNIQUE EVALUATION

FINAL REPORT

C.L. May
D.J. Spencer

APRIL 1974

Prepared for
GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

Under Contract No. NAS5-21746

Original photography may be purchased from:
EROS Data Center
10th and Dakota Avenue
Sioux Falls, SD 57198

Prepared by
TRW Systems Group
One Space Park
Redondo Beach, California

206

ATTENTION

AS NOTED IN THE NTIS ANNOUNCEMENT,
PORTIONS OF THIS REPORT ARE NOT LEGIBLE.
HOWEVER, IT IS THE BEST REPRODUCTION
AVAILABLE FROM THE COPY SENT TO NTIS.

ERTS IMAGE DATA COMPRESSION TECHNIQUE EVALUATION

FINAL REPORT

C.L. May
D.J. Spencer

APRIL 1974

Prepared for
GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

Under Contract No. NAS5-21746

Prepared by
TRW Systems Group
One Space Park
Redondo Beach, California

Table of Contents

Table of Contents (Continued)

APPENDICES

LIST OF ILLUSTRATIONS

Figure | Page

LIST OF ILLUSTRATIONS - CONTINUED

LIST OF ILLUSTRATIONS - CONTINUED

LIST OF ILLUSTRATIONS - CONTINUED

LIST OF TABLES

1. INTRODUCTION

The objective of this investigation is the determination of the degree of data compression which can be obtained for ERTS multispectral imagery using a set of algorithms which produce zero or minimal distortion in the reconstructed data. The results obtained can be used for determining the feasibility of data compression as an integral part of future ERTS programs, either for spacecraft or ground processing applications.

The investigation of data compression techniques for ERTS multispectral data has been completed and the desired results have been obtained. The study has shown that the Spectral-Spatial-Delta-Interleave (SSDI) algorithms permit an average data compression of more than 2:1 for a strictly information preserving reconstruction and 3:1 or better if a small degree of distortion is permissible in the reconstructed data. In terms of storage requirements, a 100x100 nmi scene can be compressed to a single reel of tape for a saving of three tapes per scene.

1.1 BACKGROUND OF THIS STUDY

The multispectral imaging sensors of ERTS-A generate tens of billions of bits daily. In future missions this figure will continue to increase as higher resolution sensors and additional spectral bands are added. Such volumes of data produce severe problems in communication links, in ground data processing, and in ground data storage and archiving. In 1970, TRW began an investigation of low complexity data compression techniques tailored to the characteristics of multispectral data in order to alleviate the magnitude of such data handling problems. During the in-house study, the class of techniques termed the Spectral-Spatial-Delta-Interleave (SSDI) algorithms were developed. The SSDI technique is strictly information preserving to provide reconstructed data identical to the digitized source data entering the compressor. Such a technique preserves the data and cannot be criticized by any user as invalidating his data requirements. Since strictly information preserving techniques faithfully compress all the input data, including sensor and quantization noises, the degree of compression obtained is limited. A modification of the SSDI algorithm was developed to permit a higher degree of compression at the expense of a slight controlled distortion in the reconstructed data. This essentially information preserving algorithm yields data acceptable by many users of the data since the distortion is comparable to the system noise level. Preceding the current investigation, TRW developed a set of computer programs capable of simulating these various algorithms for use with a variety of multispectral imaging data sources. These programs were validated using digital imagery obtained from the Laboratory for the Applications of Remote Sensing (LARS) C-1 flight data and Apollo 9 data (S065 experiment). The programs measure pertinent source and compressed data statistics, generate compressed and reconstructed data for the various algorithms, and reformat the reconstructed data for the subsequent generation of photographic imagery.

1.2 STUDY TASKS

The following 6 tasks were delineated in the Data Analysis Plan as necessary to meet the objectives of the investigation:

1. Modification of the master data compression computer programs which were previously developed at TRW Systems to enable reformatting the bulk MSS digital tapes, provide for selection of the desired segments of the data to be processed and generation of the appropriately formatted output products.

2. Selection of 30 subscenes and 4 full scenes to be processed based on the desired object classes and the tapes available from NASA.

3. Measurement of pertinent MSS data statistics for all scenes processed.

4. Measurement of desired global and time-varying compressed data statistics.

5. Generation of reconstructed imagery for selected full scenes, including a scene processed by an essentially information preserving algorithm and a scene subjected to simulated channel errors.

6. Evaluation and interpretation of the investigation.

In order to obtain the body of data required to accomplish these study tasks, the set of output data products given in Table 1.1 was obtained for each scene processed.

In addition to these proposed tasks, two additional tasks were added to supplement the study. First, a tape of digitized spacecraft data was obtained from NASA-GSFC and processed in the same manner as the bulk MSS tapes. This task serves to compare the compression obtained by the algorithms for both types of data. In addition, an investigation was performed concerning the hardware implementation of the SSDI/Rice algorithm for spacecraft applications.

Table 1.1. Data Measurements Obtained for Each Scene Processed

1. MSS Data Statistics

   a. Data mean and variance per band and over all bands.

   b. Cross spectral-spatial correlation.

   c. Spectral correlation (joint probability distribution function).

2. Data Compression Performance

   a. Probability distribution function of first difference obtained by the SSDI, SSDIA, and SSDIAM modes.

   b. Probability distribution function of the SHELL, SSDI, SSDIA, and SSDIAM symbols.

   c. Compression achieved by fixed Huffman coding of the scene using the SHELL, SSDI, SSDIA, and SSDIAM modes.

   d. Entropy of these distributions.

   e. Line-by-line time-varying and overall data compression using the fixed Huffman, adaptive Huffman, and Rice algorithms on the selected compression mode.

   f. Buffering statistics of the Rice code.

   g. Huffman codes associated with the various compression modes.

1.3 SUMMARY OF RESULTS OBTAINED

The investigation yielded a significant amount of data regarding source statistics, compression statistics, algorithm performance, and hardware complexity considerations. The key results are summarized below and discussed in depth in section 4 of this report.

- Compressed bit rates, averaged over the scene, vary from a minimum of 1.22 bits/sample to a maximum of 3.747 bits/sample for the strictly information preserving algorithms.

- The compressed bit rates obtained, averaged over all scenes processed, are:

  2.99 bits/sample for SHELL/global Huffman
  2.98 bits/sample for SSDI/global Huffman
  2.92 bits/sample for SSDIA/global Huffman
  2.50 bits/sample for SSDIAM/global Huffman
  2.67 bits/sample for SSDI/adaptive Huffman
  2.70 bits/sample for SSDI/Rice

- For well-behaved data, the SSDIA technique gives a lower compressed bit rate than the SSDI algorithm. For anomalous data such as that produced by a defective sensor, this is not always true.

- The essentially information preserving SSDIAM produces a significantly lower compressed bit rate than the strictly information preserving SSDIA algorithm. The effects of such distortion appear minimal when properly performed and areas of high detail are well preserved with no slope overload or overshoot.

- The strictly information preserving algorithms can compress four full 100x100 nmi scenes to occupy the same number of magnetic tapes currently required to store one full scene. An even greater reduction is possible with the essentially information preserving algorithms.

- The effect of channel errors is minimal if the channel bit error rate is at least 10^{-6}. Channels with higher error rates can be used if frequent memory updates are included.

- An implementation of the SSDI/Rice algorithm was developed to illustrate the feasibility of operation at rates above 100 Megabits/second with moderate complexity. Parallel data compressor units operating on blocks of data permit operation at several hundred Megabits/second.

- The SSDI/Rice algorithm is well suited for spacecraft data compression applications. The SSDI/Huffman algorithm provides an efficient data compression and reconstruction technique suitable for use in ground applications.

2. TECHNICAL DISCUSSION OF WORK PERFORMED

Various data compression algorithms were exercised in this study on a variety of multispectral data sources. The results and conclusions of these studies are presented below in Sections 3 and 4. This section describes these algorithms and the analytic/computational tools developed in order to provide a framework for the discussion of the results obtained.

2.1 DESCRIPTION OF DATA COMPRESSION ALGORITHMS USED

2.1.1 Spectral-Spatial-Delta-Interleave Algorithm

The Spectral-Spatial-Delta-Interleave (SSDI) algorithm is a method of data compression, developed for multispectral data, which removes a maximum amount of redundancy subject to the constraints of minimizing complexity and maximizing operating speed. This compression algorithm first operates on the spatial redundancy in each spectral band and then uses the information obtained to reduce spectral redundancies between adjacent bands.

In order to provide a conceptual description of the basic SSDI algorithm and several of its modifications, a situation in which there are three spectral bands, α, β, γ will be described. Each ground picture element (pixel) I consists of three quantized spectral components, I_α, I_β, and I_γ. Figure 2.1-1 may be helpful in visualizing the quantities involved.

The algorithm proceeds in the following fashion. First, within each spectral band, each pixel intensity is subtracted from the intensity preceding it in the scan direction. (This technique is essentially DPCM, treating each spectral band separately.) To each pixel, then, there can be assigned a triple of these differences denoted $(\Delta\alpha, \Delta\beta, \Delta\gamma)$.

Next, these "deltas" are themselves differenced to obtain second differences in adjacent spectral bands; viz $d_A = \Delta\beta - \Delta\alpha$ and $d_B = \Delta\gamma - \Delta\beta$. Here too, each pixel may be assigned the triple $(\Delta\alpha,\ d_A,\ d_B)$ which provides the same information as the triple $(\Delta\alpha,\ \Delta\beta,\ \Delta\gamma)$. However, due to spectral band correlation it should be true that on the average, $|d_A| + |d_B| \leq |\Delta\beta| + |\Delta\gamma|$, and the d_A and d_B are clustered closer to the origin than the first differences $\Delta\beta$ and $\Delta\gamma$.

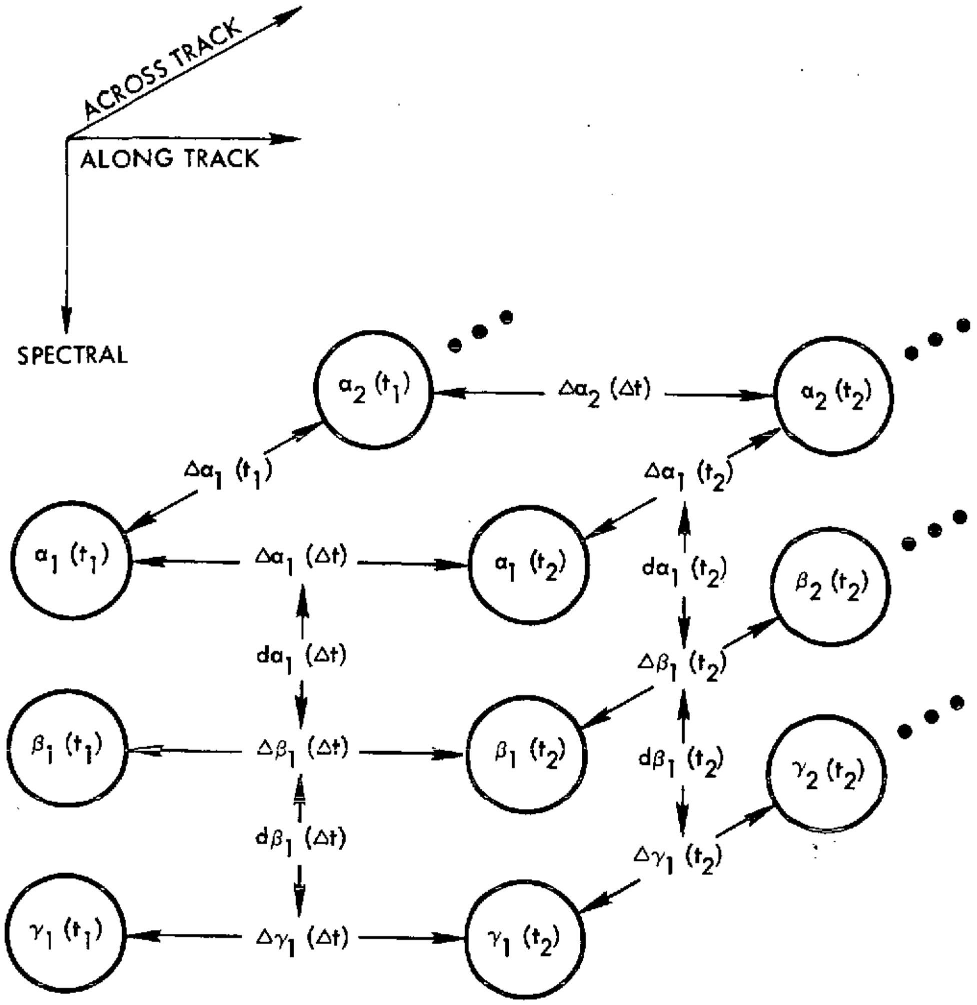


Figure 2.1-1. Definition of First and Second Order Differences in SSDI

These differences are transmitted in a manner allowing the original PCM sensor data to be recovered exactly from the coded sequence. Corresponding to each pixel, the triple $(\Delta\alpha, d_A, d_B)$ is developed. Given the preceding pixel intensities, denoted $(I_\alpha^{i-1}, I_\beta^{i-1}, I_\gamma^{i-1})$ the current intensities may be obtained by the recursion relationships

$$I_{\alpha}^{i} = I_{\alpha}^{i-1} + \Delta\alpha$$

$$I_{\beta}^{i} = I_{\beta}^{i-1} + \Delta\alpha + d_A$$

$$I_{\gamma}^{i} = I_{\gamma}^{i-1} + \Delta\alpha + d_A + d_B$$

The final step in the SSDI algorithm is the encoding of these differences. During this study three methods of encoding were investigated for use with the basic SSDI algorithms. They are 1) global Huffman coding; 2) adaptive Huffman coding; and 3) Rice encoding. These coding algorithms are described in Sections 2.2 and 2.3.

2.1.2 SSDIA — A Block Averaging Extension to SSDI

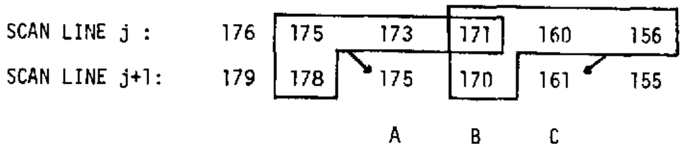
Most data — including the test data used in this study — contain sensor and sampling noise. If this noise could be reduced, a higher compression rate could be obtained. Two modifications of the SSDI algorithm have been developed to ameliorate the noise problem. The first modification is called SSDIA to denote that pixel averaging is employed. SSDIA is discussed in this section. The second modification — SSDIAM — is discussed in Section 2.1.4. The SSDIA, like the SSDI, is a strictly information preserving algorithm while the SSDIAM permits a degree of controlled distortion in the reconstructed data.

The SSDIA algorithm is based on the observation that sensor noise is essentially uncorrelated from pixel to pixel. This fact degrades SSDI compression since the differential magnitudes can be large when noise on one pixel is positive while noise on the preceeding pixel has a negative value. On the other hand, the effects of the noise can be reduced by differencing the current pixel value in each spectral band with the average value of Q preceding pixel values in the same band since the (uncorrelated) noise will increase the value of some of these Q pixels while decreasing the value of others. If these Q pixels are contiguous to the present pixel, a high degree of correlation should exist with the mean of these adjacent pixels and the current pixel intensity. An example of the SSDIA algorithm is provided in Figure 2.1-2, based on the use of four previously transmitted pixel intensities. No future pixel values are used in the mean evaluation because of the difficulty that would result in the decoding process. In general, the mean, μ, of Q preceding pixels would be a weighted sum of these pixels with the weightings a function of inter-pixel correlations.

$$\mu_{\alpha}(j) = \sum_{i=1}^{Q} a_i I_{\alpha}(i) \qquad \Delta\alpha = I_j - \mu_{\alpha}(j)$$

Thus, SSDIA increases the compression ratio by the averaging of pixels whose intensities are correlated with 1_j but whose noise components are effectively uncorrelated with 1_j. Another benefit derived from the SSDIA involves a decrease in the magnitude of the first differences when a large step in intensity, such as is produced by an edge in the scanned image, occurs between two adjacent pixels in the scan line. Such a situation is illustrated in Figure 2.1-2 between pixels B and C. Note that for the SSDIA the second differences, d_A and d_B, are obtained in the same manner as for the SSDI algorithm.

PIXEL INTENSITIES



SCAN LINE j :	176	175	173	171	160	156
SCAN LINE j+1:	179	178	175	170	161	155
			A	B	C	

	PIXEL A	PIXEL B	PIXEL C
Block Mean (to 8 bits)	174	170	164
$\Delta\alpha$,SSDI	-3	-5	-9
$\Delta\alpha$,SSDIA	+1	0	-3

Pixel intensities are given for two scan lines in spectral band α. The averaging will be performed over a set of four previously computed adjacent pixels and an equal weighting of 1/4 is given each intensity. The selected set for pixels A and C are enclosed in boxes. The compression improvement for these pixels by averaging is given above.

Figure 2.1-2. Illustration of the SSDIA Averaging Technique

2.1.3 Shell Coding

Another form of coding the triples has been investigated. This method is referred to as shell coding for reasons which will become clear. Instead of coding each differential component independently, the triple of differences is encoded jointly. That is, the triple of differences is mapped into a scalar quantity.

As illustrated in Figure 2.1-3, each pixel corresponds to a point in a three-dimensional vector space with coordinates equal to the intensities of each of the three spectral bands. The vector space is assumed to be quantized into cells. If each intensity is quantized into seven bits, there are 21 cells in this "spectral" vector space. Therefore, each pixel could be completely described by numbering each cell and assigning to each pixel the number of the cell which contains the vector tip. This is essentially what is done in ordinary PCM where the quantized cell coordinates are projected onto the intensity axis and transmitted in sequence. Differential PCM accomplished this by placing a "floating" cube centered on the cell containing the last pixel intensities, as shown in Figure 2.1-4. The new set of pixel intensities is then mapped onto the coordinates of the "floating" cube as differential information.

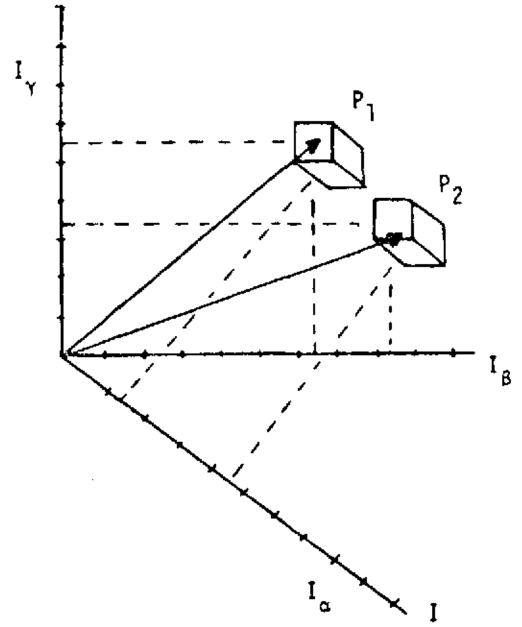


Figure 2.1-3. Cells Correspond to Two Picture Elements, P_1,P_2

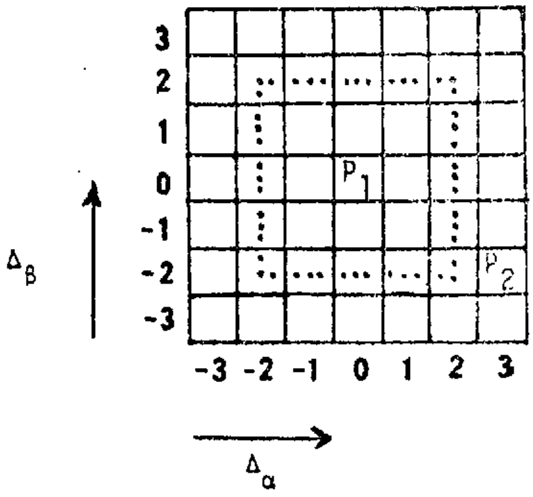


Figure 2.1-4. Floating Cube Designati of P_1 and P_2. Dotted Li Designates Shell 2 (L=2) in Two Dimensions

A more sophisticated method of transmitting this same information is the transmission of only a single (digital) number assigned to the particular cell of the cube within which the differential intensity vector lies. However, if the total number of cells (in any direction away from the center cell) need to handle the differential vector is q in mangitude the total number of cells which must be labeled is 2^{6q}. For M spectral bands, the number of cells Q lying on a shell a distance L away from the origin (0,0,...,0) is

$$Q = (2L + 1)^M - (2L - 1)^M, \text{ for } L \geq 1$$

Shell coding transmits the actual cell label by conveying two pieces of information; the shell of the cube on which the differential vector lies and the label of the cell in that shell which contains the vector. This technique has the potential for achieving good data compression since three quantities are mapped into one.

The average length of the shell code depends on the probability density of the number of triples lying within each shell. If most levels are concentrated within the first few shells and the maximum number of shells required is quite large, then significant data compression is possible. (The shell probability distribution appears similar to an X^2 distribution.) The average length of the shell code depends upon the shell probabilities. Since very few triples (0,0,0) occur, almost all sequences have at least five bits per triple. In addition, the number of the shell must also be transmitted, thereby increasing the code length. The shell levels are Huffman coded according to the shell distribution.

Neglecting the additional information required to specify the shell number (L), the following is a comparison of the number of bits required to compress a triple of differential information with shell encoding versus that required for the SSDI code. The first column gives the largest (in absolute amplitude) component of the triple, the second column gives the number of bits required to specify the cell in that shell containing the triple, and the last column gives the total length of the SSDI code required to transmit the triple.

Note that the third column contains a range for the total number of bits required. This is because the total SSDI code length is variable depending on what the other two levels in the triple are for a given maximum level. As an example (3, -3, 3) requires 18 bits total while (0, 3, 0) requires only 8 bits in the SSDI even though they both lie on the third shell.

Max. Level Per Triple	No. Bits For Shell	No. Bits For SSDI
-4	9	
-3	8	8 - 18
-2	7	7 - 15
-1	5	5 - 9
0	1	3
1	5	4 - 6
2	7	6 - 12
3	8	8 - 18
4	9	

Again, as illustrated by the results given in this tabulation, the average length of the shell code depends upon the shell probabilities. Since very few triples (0, 0, 0) occur, almost all sequences have at least five bits per triple. In addition, the shell number L, must also be transmitted which increases the code length.

The SSDI and Shell algorithms are two methods of source coding the triples ($\Delta\alpha$, d_A, d_B) and each is inherently strictly information preserving but can be easily extended to be essentially information preserving. The SSDI source encodes the data using a Huffman code based on the statistical occurrence of the differentials considered singly while the shell code uses the shell statistics to jointly encode the triple of the differentials.

2.1.4 SSDIM and SSDIAM — Essentially Information Preserving Algorithms

The SSDIM and SSDIAM algorithms allow the mapping of data intensities within specified limits in a fashion which increases the compression obtained while holding the distortion in the reconstructed data to controlled levels.

These essentially information preserving (eip) algorithms utilize a simple pre-processing of the data to decrease the average magnitude of the SSDI or SSDIA symbols and tends to average out sensor and quantization noise effects in the data.

The mapping is performed on the original data sample intensities so that the resulting reconstructed data cannot deviate from the original data by more than m quantization levels, where m is a level specified by the user. Values of m = 1 or 2 are useful for eliminating much of the deleterious effects of sensor and quantization noise on the compression algorithms without producing noticeable visual degradation to the reconstructed image while higher values produce visual changes with severity depending on value m and scene content. Section 3 will discuss this situation in more detail. In all cases, no intensity element in the reconstructed data has an error of more than m quantization levels and the mean square error is always less than $m^2/2^{2q}$ of the maximum dynamic range, where q is the number of bits per data sample.

The SSDIM mapping is performed over a block of k pixels at a time in the following fashion. First, the integer-block average of all pixels is formed within each spectral band. Second, the individual block intensities are shifted in level toward that mean value, with a maximum shift of m levels up or down. If the intensity lies at the block mean, its value is unchanged. Following this operation the conventional SSDI operation is formed on the mapped intensities. Note that this averaging and mapping operation always decreases the sum of magnitudes S_T of the SSDI symbols, implying a decrease in variance of the symbol probability distribution function and a corresponding increase in compression.

An example of SSDIM operation serves to illustrate the averaging and mapping process. Assume that the original intensity values in the (3) spectral bands are

α	10	17	15	$\mu_\alpha = 14$
β	12	10	12	$\mu_\beta = 11$
γ	5	10	14	$\mu_\gamma = 10$

With a mapping of m = 1, the following intensities are established

α	11	16	14
β	11	11	11
γ	6	10	13

With a mapping of m = 4, the following intensities result

α	13	13	13
β	11	11	11
γ	9	10	10

Based on these three sets of intensities we get the following sequence of symbols and corresponding values of S

$$\text{SSDI} = 7,\ -9,\ 7,\ -2,\ 4,\ 2 \qquad \rightarrow S_\tau = 31$$

$$\text{SSDIM}\big|_{m=1} = 5,\ -5,\ 4,\ -2,\ 2,\ 3 \qquad \rightarrow S_\tau = 21$$

$$\text{SSDIM}\big|_{m=4} = 0,\ 0,\ 1,\ 0,\ 0,\ 0 \qquad \rightarrow S_\tau = 1$$

The SSDIAM is formed in a similar fashion but the mapping average is performed over the block of pixels in both the current scan line and those in the preceding scan line. Following the mapping, the conventional SSDIA operation is performed on these mapped intensities. While the actual reconstructed data may differ depending upon whether the SSDI or SSDIAM operation is used, the distortion bound remains the same. The SSDIAM normally produces a somewhat higher compression than the SSDIM.

The averaging can be performed over fixed blocks or the block size can vary adaptively with changing scene characteristics. Fixed block sizes permit an algorithm of lower complexity but generally provides less compression than adaptive versions. As the block size increases to a degree where the block mean differs substantially from the means of subblocks, performance deteriorates. An adaptive block begins at some minimal block size L and increases the size, one element at a time, until the mean begins to differ too widely from the block mean μ_L. For each additional intensity added, the new block mean can be calculated recursively from the last as

$$\mu_{L+1} = \left(\frac{L}{L+1}\right)\ _{L} + \frac{I_{L+1}}{L+1} .$$

If $|\mu_{L+K}|/|\mu_L|$ falls outside prescribed bounds, the block size is truncated to include L + K - 1 pixels. The block size increases in regions of low data activity and decreases in regions of high data activity. In any case, whether SSDIM or SSDIAM, these mapping techniques avoid problems of overshoot and slope overload. Use of the SSDIM or SSDIAM does not entail any modification in the reconstruction algorithms.

2.2 HUFFMAN SOURCE CODING

Several algorithms exist for efficiently coding sources whose statistics are known. These techniques have been investigated at TRW and the Huffman code was chosen as being the most desirable algorithm for ground processing. The Huffman code has all the properties required to ensure unique decoding with the minimum number of bits, coding each symbol at a time. Furthermore, it permits use of a "table look-up" decoding algorithm which can be performed rapidly.

A difficulty encountered in practical applications is the cumbersome algorithm required for the classical synthesis of a Huffman code given the statistics of the source symbols S. TRW has developed a more efficient technique for generation of Huffman codes. The latter algorithm also permits grouping of low probability symbols together for simplified decoding. Following a discussion of the classical Huffman code synthesis, the new algorithm will be described.

2.2.1 The Classical Synthesis of Huffman Codes[1]

To explain the classical Huffman code synthesis, consider a source S^1 with symbols s_1^1, s_2^1, ..., s_q^1 and symbol probabilities P_1^1, P_2^1, ..., P_q^1. Without loss of generality, it may be assumed that the symbols are ordered so that $P_1^1 \geq P_2^1 \geq \ldots \geq P_q^1$. The two least probable symbols S'_{q-1} and S'_q may be combined and thought of as a single symbol with probability equal to $P_{q-1}^1 + P_q^1$. Thus, a new source S^2 may be constructed with symbols s_1^2, s_2^2, ..., s_{q-1}^2 and probabilities P_1^2, P_2^2, ..., P_{q-1}^2, again ordered so that $P_1^2 \geq P_2^2 \geq \ldots$

$\geq P^{2}_{q-1}$. The reduction can then be repeated to obtain a sequence of sources $S^1, S^2, \ldots, S^{q-1}$ where S^{q-1} has only two symbols.

A compact instantaneous binary code for the final reduction S^{q-1} is the trivial code with words 0 and 1. Working backward from this final reduction, the Huffman code is inductively synthesized as follows:

Assume that a compact instantaneous code has been found for the source S^i. One of the symbols, say s^i_α, is formed from the two least probable symbols of S^{i-1}. These two symbols are S^{i-1}_{q-i} and S^{i-1}_{q-i+i}. Each of the other symbols of S^i correspond to one of the remaining symbols of S^{i-1}. The code for S^{i-1} is formed from the code for S^i thus:

To each symbol of S^{i-1}, except s^{i-1}_{i-q} and s^{i-1}_{i-q+1}, assign the codeword used by the corresponding symbol in S^i. The codewords assigned to S^{i-1}_{i-q} and S^{i-1}_{i-q+1} are formed by adding a 0 and 1, respectively, to the codeword used for S^i_α. An example of the synthesis procedure for a given source is illustrated in Figure 2.2-1.

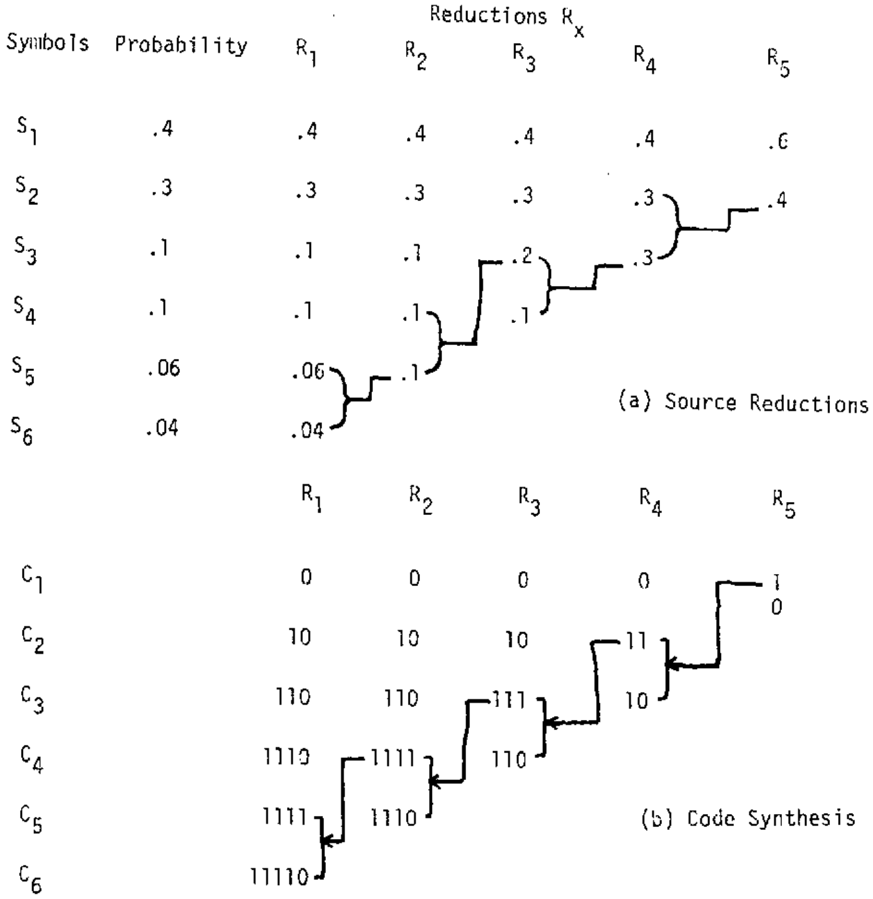


Figure 2.2-1. Classical Huffman Code Synthesis

Each symbol s_i of the source S^1 is thus assigned a codeword of length l_i. The average code length for this source is therefore

$$\overline{L} = \sum_{i=1}^{q} P_i^1 \ell_i$$

where $\overline{L}$ satisfies the inequality

$$0 \le \overline{L} \le H = -\sum_{i=1}^{Q} P_i^1 \log P_i^1$$

where H is the entropy of the source S^1.

The difficulty imposed by the classical Huffman synthesis arises in the forward flow of the code generation between successive reduced sources. This procedure is very inefficient with respect to both storage and time when used as the basis of a computer algorithm for coding a source.

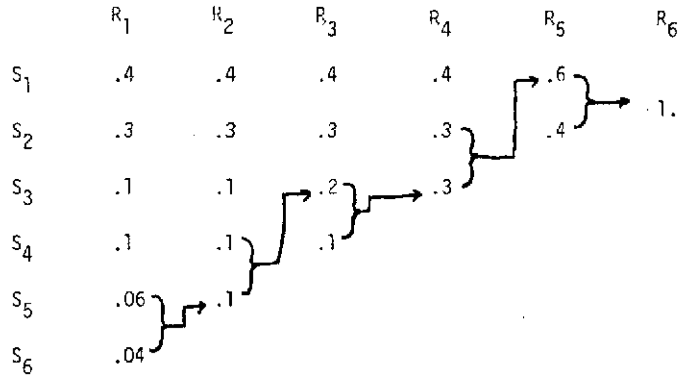
2.2.2 An Improved Huffman Algorithm for Computers

The new algorithm separates the source reductions from the code synthesis. The first part of the algorithm keeps track of the number of times each symbol in the original source is grouped during the sequence of source reductions. This contains all information about the length of the codeword assigned to that symbol in the resulting Huffman code. The second part of the algorithm uses these lengths, ℓ_i, to generate a Huffman code C for the source S.

Note that the resulting Huffman code may or may not be identical to the code generated by the classical synthesis procedure. Nonetheless, the average code length is identical. Using the classical technique, many different Huffman codes can also be generated, depending on the assignment of 0 and 1 in each reduced source.

The method begins with the same source reductions as in the classical Huffman code synthesis described above with the addition of a final reduction to source S^q containing only one word. The code lengths may be determined as follows:

If the symbols s_j^i and s_{j-1}^i are combined to form s_α^{i+1} the symbol s_α^{i+1} may be considered a reduction of each of the symbols s_j^i and s_{j-1}^i. Assign to each symbol s_i^1 the length l_i, initialized to zero. Each time a symbol, or any of its subsequent reductions, is reduced the value of l_i is incremented by one. It is perhaps easier to understand this algorithm by referring to the example in Figure 2.2-2.



	R_1	R_2	R_3	R_4	R_5	R_6
S_1	.4	.4	.4	.4	.6	1.
S_2	.3	.3	.3	.3	.4	
S_3	.1	.1	.2	.3		
S_4	.1	.1	.1			
S_5	.06	.1				
S_6	.04					
ℓ_1	0	0	0	0	0	1
ℓ_2	0	0	0	0	1	2
ℓ_3	0	0	0	1	2	3
ℓ_4	0	0	1	2	3	4
ℓ_5	0	1	2	3	4	5
ℓ_6	0	1	2	3	4	5

Figure 2.2-2. Determination of Code Word Lengths, ℓ

The second part of the algorithm is illustrated in Figure 2.2-3. This part of the algorithm operates as follows:

1. The lengths of ℓ_i are ranked in the order of increasing length.
2. Symbol S_k of minimum length, ℓ_k, is assigned ℓ_k zeros.
3. Each successive symbol S_m has a code formed as

$$C_m = (C_{m-1}+1) + (\ell_m-\ell_{m-1}) \text{ zeros.}$$

SYMBOL	LENGTH, ℓ_i	OPERATION	CODEWORD
S_1	1	0	0
S_2	2	0 + 1 and 1 shift	10
S_3	3	10 + 1 and 1 shift	110
S_4	4	110 + 1 and 1 shift	1110
S_5	5	1110 + 1 and 1 shift	11110
S_6	5	11110 + 1 and no shift	11111

Figure 2.2-3. Huffman Code Synthesis Using Code Word Lengths ℓ_i

This algorithm is very fast and essentially separates the problem of code generation from that of source reduction. The only information which need be stored from the source reduction portion of the algorithm is the vector of code lengths.

2.2.3 Low Probability Symbol Grouping

Often the total number of symbols i in source S^i is quite large and many of these symbols have probabilities of a small fraction of one percent. To save time in the encoding/decoding process at the expense of a small increase in average code length, these low probability symbols can be lumped into a single symbol. As an example, after ordering symbols with decreasing probability of occurrence, the first J symbols are directly coded, where

$$\sum_{i=1}^{J} P_i \geq .99$$

The remaining symbols, having a total probability P_{J+1} of one percent or less, are grouped into symbol S_{J+1}. If M symbols are lumped into S_{J+1}, R bits must be used to describe these M symbols, where $R = \{\log_2 M\}$.* During transmission, codeword C_{J+1} is followed by R bits to describe which of the M symbols occurred. The average code length is lengthened by such a grouping by less than $P_i R$.

The advantage of grouping symbols which seldom occur is that the maximum length of any code word can be held to some predetermined length N. This simplifies the decoding algorithm and keeps the length of the required look-up table to length 2^N. These advantages in decoding are obtained at the possible expense of a slightly increased average code length.

*{ } means next larger integer.

During the encoding process, whenever one of the grouped symbols occurs the compressor transmits the sequence of bits forming code word C_{J+1} followed by R bits to describe which grouped symbol occurred. When the decoder encounters code word C_{J+1}, it uses the next R bits to decode this grouped symbol.

2.2.4 Computer Program

A computer program has been developed and tested which accepts an array of symbols and generates the Huffman code. The program allows the operator to group symbols if desired and generates the grouped Huffman code and the average bit rate if R bits are used to separate the lumped symbols.

The flowchart describing the program is given in Figure 2.2-4. The inputs required are the source symbols S_i, their associated probabilities P_i, and the maximum codeword length acceptable N. The program outputs the Huffman coded Table HUF, which contains the coded bit stream C associated with the source S.

Two major subroutines are used in this program. Subroutine ORDER reorders the symbols and their probabilities in a decreasing order so that the most probable symbols are at the top of an array O. Subroutine GROUP adds the two least probable symbols in the array O to form a source reduction. This subroutine also keeps count of the number of source reductions performed and keeps track of the original source symbols which have been combined to form each reduced symbol. Each symbol is given a bit position in an array V. If symbols s_1, s_3 and s_5 have been combined in a source reduction, that reduced symbol is represented in V as the binary word (. . . . 1 0 1 0 1). This representation allows a compact designation of groupings at each stage in the reduction.

In operation, the program takes the array of input symbols and their probabilities, calls ORDER to rank them, and combines the M least probable symbols to form the grouped symbol s_{J-M+1} of probability $P_{J-M+1} = \sum_{i=M}^{J} P_i$ (assuming $P_1 \geq P_2 \geq \cdots \geq P_{J-1} \geq P_J$). This new set of J-M+1 symbols forms the input to the basic algorithm in which successive calls to subroutines GROUP and ORDER generate successive source reductions until only two reduced symbols remain. At each stage of the reduction, array LENGTH is updated by one for

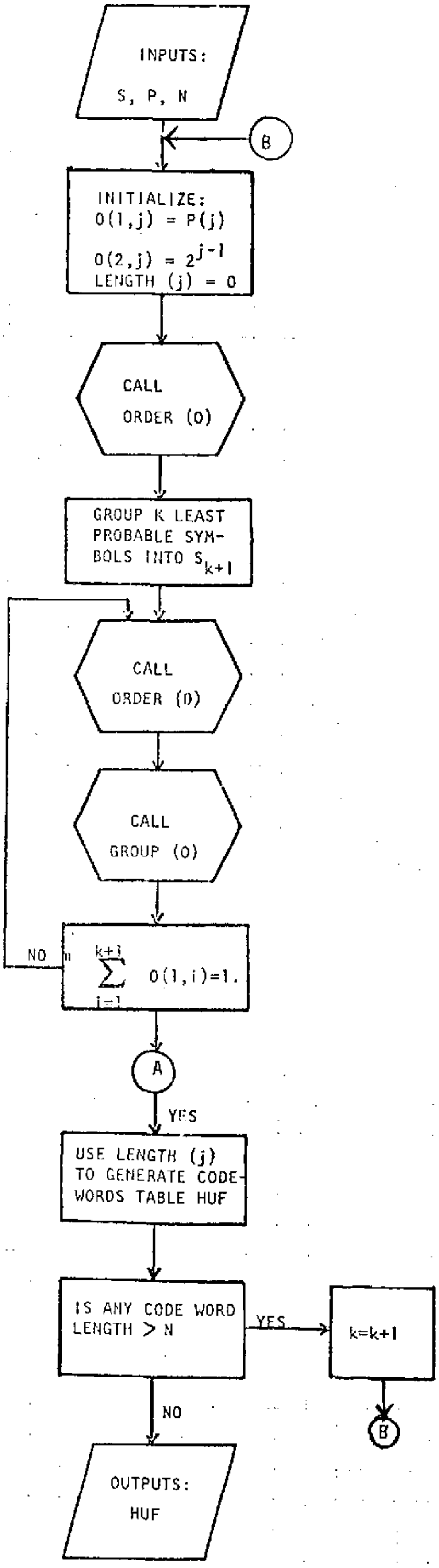


Figure 2.2-4. Program for Huffman Coding

each symbol in S which has been combined to form one of the reduced symbols which have been grouped in that step.

Following the reduction process, the array LENGTH is used to compute the binary codeword associated with all of the J-M+1 non-grouped source symbols. LENGTH is re-ordered so that the most probable symbols which have the shortest code lengths are at the top of the array. A test takes place after LENGTH is re-ordered. If the longest codeword exceeds N bits, more source symbols are grouped and the source reductions performed again until the maximum codeword length is N or less. With 256 source symbols, such an occurrence is guaranteed at some stage of grouping.

The generation of the codes then begins with the minimum length codeword and proceeds from word to word with the successive steps of adding 1 to the previous codeword and adding the required number of zeros to fill the word.

Table HUF is then generated where all entries corresponding to non-grouped symbols contain the computed Huffman codeword. For all grouped symbols, the entry in HUF contains the lumped prefix codeword C_{J-M+1} followed by 8 bits giving the symbol directly.

2.2.5 Adaptive Huffman Coding

The adaptive Huffman algorithm used in the TRW simulation program produces a new Huffman code for each scan line of data based on the statistics of the difference symbols generated for the preceding scan line. Two concurrent operations are therefore performed for the processing of a given scan line, the symbols for this line are encoded using the Huffman code developed based on the statistics of the previous line, and the probability distribution of symbols generated for the current line is computed. At the end of the scan line the Huffman coding subroutine is called and the new code is computed and stored. The same technique is used for reconstruction since the decoding processor has regenerated the previous line of symbols and can develop the same Huffman code.

This technique is very rapid and does not require significant storage for performing the required operations. The efficiency of the code generated depends upon the correlation of symbol statistics from one scan line to the next. Generally, the match of statistics is quite good and certainly superior to a technique using the symbol statistics from one block of data on a scan line to develop a code for use in the succeeding block of data on that line.

This form of adaptive coding yields excellent results in subscene processing since the scan lines are relatively short and the subscene normally contains a single object class with rather uniform symbol statistics. A more efficient coding for large scenes might require breaking each scan line into several segments and computing a separate Huffman code on each segment for use in encoding symbols on the corresponding segment of the following scan line.

2.3 THE RICE CODING ALGORITHM

The Rice encoding algorithm, developed by R. F. Rice at JPL[2] is a variable length coding system which is basically strictly information preserving. Operating on a sequence of source symbols, the Rice machine adapts by selecting one of three coding schemes with computational capability for optinally switching to that one of three codes which is compatible with the data activity. Code $\overline{FS}$ performs well with low data activity, code FS performs well for data of medium activity, and code CFS performs best with very active data. In order to adapt to rapid changes in activity, the basic Rice compressor monitors data activity and selects the appropriate code mode based on small blocks of data symbols.

The resulting coding system produces output rates within .3 bits/sample of the one-dimensional entropy of the samples and cannot expand the data by more than .1 bits/sample under any circumstance. While the Rice machine can operate on any source of data, the input to the Rice encoder, as described here, is the sequence of SSDI, SSDIA, or SSDIAM symbols.

Rice assumes that the adjacent samples of Δ are statistically independent and that the probability distributions of these symbols decrease monotonically on either side of $\Delta = 0$. For his assumed zero-memory source, Rice seeks to assign the shortest code words to source symbols which have the greatest probability of occurrence and the longest code words to those symbols which have the least probability of occurrence. This is the same idea underlying the Huffman code as described in Section 2.2.

For each block of J symbols, the entropy of first order linear differences is given by

$$H(P) = \sum_{i=1}^{q} p_i \log p_i \qquad \text{bits/pixel}$$

where p_i represents the probability of the $i^{\underline{th}}$ source symbol and the log function is base 2. Parameter q can vary from zero to $2^{N+1}-1$, where N is the number of bits used for source quantization. Entropy can be considered as the quantitative measure of the source data activity.

L(P) is the number of bits/pixel required to code the sequence of difference samples have the distribution $P = \{p_i\}_{i=1}^{2^{N+1}-1}$. Under Rice's assumption of a zero memory source, the average code length cannot be less than H(P):

$$E[L(P)] \geq H(P)$$

where E denotes the expectation operator.

The generality of Rice's model assumes that P can change completely from block to block and his coding algorithm can change from block to block, depending on the distribution, P, within each block. He measures the system performance by comparing E[L(P)] with the lower bound H(P). In operation, the Rice algorithm monitors the data activity of each block of symbols and select one of three codes, dependent on the activity range.

2.3.1 The Fundamental Sequence and Code Assignments

Let Z_j represent the $j^{\underline{th}}$ difference sample in a block of J transformed symbols. This input sequence appears as a sequence of symbols S_j where each input sample Z_j is associated with some symbol S_i by the following assignments.

$$0 \longleftrightarrow S_1$$

$$+1 \longleftrightarrow S_2$$

$$-1 \longleftrightarrow S_3$$

$$\begin{array}{ccc}
+2 & \longleftrightarrow & S_4 \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
+127 & \longleftrightarrow & S_{254} \\
-127 & \longleftrightarrow & S_{255}
\end{array}$$

Following Rice[2], a q x J sample matrix can be constructed with elements a_{ij} given by

$$a_{ij} = \begin{cases} 1 & \text{iff } Z_j = S_i : \begin{array}{l} i = 1, 2, \ldots, q \\ j = 1, 2, \ldots, j \end{array} \\ \\ 0 & \text{otherwise} \end{cases}$$

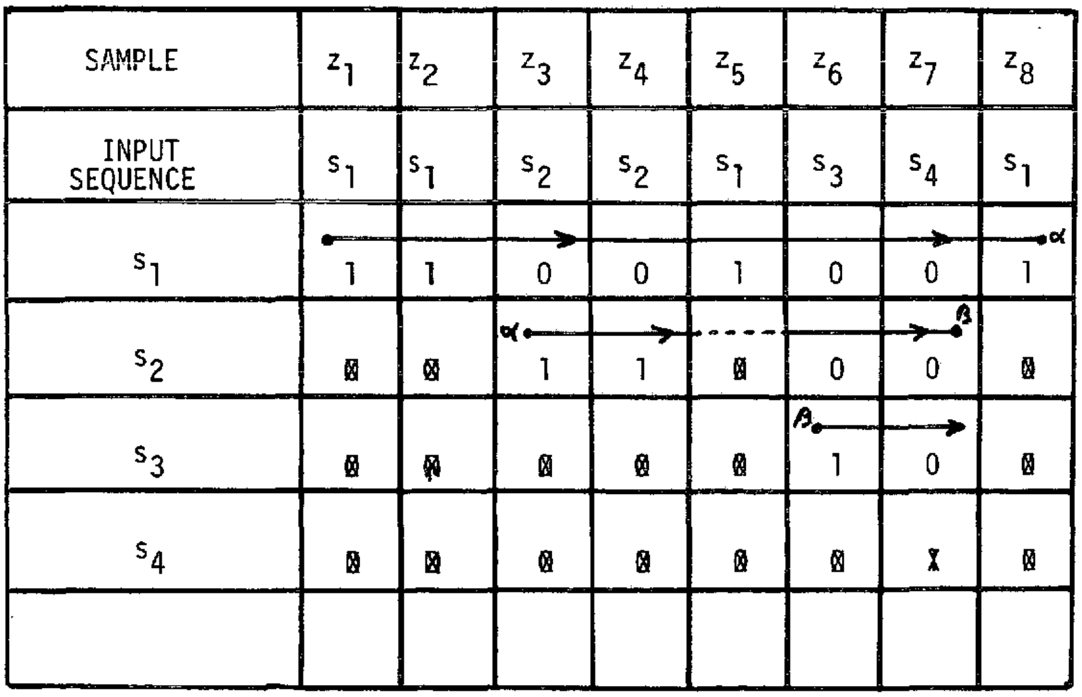
This is illustrated in Figure 2.3-1 for q = 4, J = 8 and an assumed input sequence $S_1S_1S_2S_2S_1S_3S_4S_1$. The fundamental sequence FS is generating by a "wiggle" operation involving three steps:

1. Cross out all zeros which lie below a 1 in the sample matrix.
2. Cross out any remaining 1's in the last row.
3. Letting r_i denote the residual 0's and 1's remaining in the i^{th} row, concatenate the $\{r_i\}$ to form the FS. This operation is illustrated in Figure 2.3-2 for the example given in Figure 2.3-1 to produce the fundamental sequence.

$$FS = r_1r_2r_3 = 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0.$$

SAMPLE	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
ASSUMED INPUT SEQUENCE	s_1	s_1	s_2	s_2	s_1	s_3	s_4	s_1
s_1	1	1	0	0	1	0	0	1
s_2	0	0	1	1	0	· 0	0	0
s_3	0	0	0	0	0	1	0	0
s_4	0	0	0	0	0	0	1	0

Figure 2.3-1. Sample Matrix



SAMPLE	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
INPUT SEQUENCE	s_1	s_1	s_2	s_2	s_1	s_3	s_4	s_1
s_1	1	1	0	0	1	0	0	1
s_2	~~0~~	~~0~~	1	1	~~0~~	0	0	~~0~~
s_3	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	1	0	~~0~~
s_4	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	X	~~0~~

Figure 2.3-2. Fundamental Sequence Generation
FS = 11001001110010

While not necessarily the best code for the FS, the code table used by Rice is given in Table 2.3-1. Code word designation ℓ_1 corresponds to the i^{th} word of a variable length binary code. As shown in the table, an 8-word variable length code is used. Coding the FS (or its complement) means assigning one of the code words in Table 2.3-1 to each sub-sequence of three binary digits making up the FS. Each sub-sequence of three bits is an address to an 8-word table containing the code. Rice's code word assignment is given in Table 2.3-2.

Table 2.3-1. Variable Length Code

Code Word Designation	Actual Code Word
ℓ_1	0
ℓ_2	1 0 0
ℓ_3	1 0 1
ℓ_4	1 1 0
ℓ_5	1 1 1 0 0
ℓ_6	1 1 1 0 1
ℓ_7	1 1 1 1 0
ℓ_8	1 1 1 1 1

Table 2.3-2. Code Word Assignment

Address	Code Word Assignment
0 0 0	ℓ_1
0 0 1	ℓ_2
0 1 0	ℓ_3
1 0 0	ℓ_4
0 1 1	ℓ_5
1 1 0	ℓ_6
1 0 1	ℓ_7
1 1 1	ℓ_8

As an example, use the FS previously determined. A dummy zero is added at the end of the FS so that the FS is divisible by 3. CFS and $\overline{CFS}$ are given here

FS:	(110)	(010)	(011)	(100)	(100)
CFS:	(11101)	(101)	(11100)	(110)	(110)
$\overline{FS}$:	(001)	(101)	(100)	(011)	(010)
$C\overline{FS}$:	(100)	(11110)	(101)	(11100)	(101)

In this example, FS is of length 15 bits, CFS is of length 19 bits, and $C\overline{FS}$ is of length 19 bits.

Letting F be the length of the block FS normalized to bits/sample, the decision as to mode used is given by

$C\overline{FS}$ if $1 \le F < 1.5$
FS if $1.5 \le F < 3$
CFS if $F \ge 3$

Figure 2.3-3 illustrates the basis of these range selections and shows how close each coding mode lies to the entropy curve. For the example sequence of eight symbols, $F = 15/8$ implying that FS should be transmitted.

2.3.2 Split-Pixel Modes

An additional set of operating modes can be added to the basic compressor to extend efficient performance to data sources with entropy outside the principal operating range: i.e., very active data.

The basic compressor operates independently of the level of quantization and can be made to treat m-bit data as n-bit data ($m \ge n$) simply by shifting out the m-n least significant bits of each data sample. The split-pixel option allows the basic compressor to compress only the n most significant bits of each m-bit sample and to transmit back directly the most significant of the remaining k=m-n bits. Each such split-pixel mode is designated by the notation (n,k). Note that if $n+k < m$, distortion can occur when the data is reconstructed. Thus, the Rice algorithm can be extended to become essentially information preserving through such an operation of the split-pixel modes.

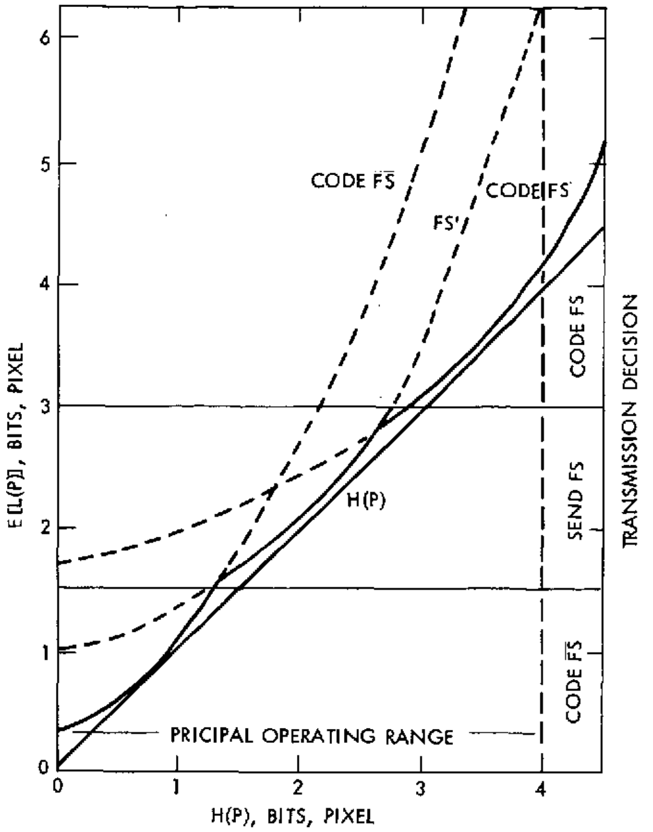


Figure 2.3-3. Dynamic Performance Curves for $0 \leq H \leq 4$

Split-pixel performance versus source entropy is given in Figure 2.3-4. As shown, one of the (n,k) modes lies close to the entropy line for $4 \leq H < 8$. By computing the entropy of the block before encoding it, the decision as to which, if any, split-pixel mode to use can be made. Alternatively, since adjacent scan lines are normally highly correlated it is very likely that the particular (n,k) mode best for one line would also be best for the next. This technique, used in the TRW simulation, computes the entropy of each scan line to determine the best (n,k) mode and uses that mode for encoding the next line. In general, this technique comes close to optimality and is definitely easier to implement than the first technique mentioned.

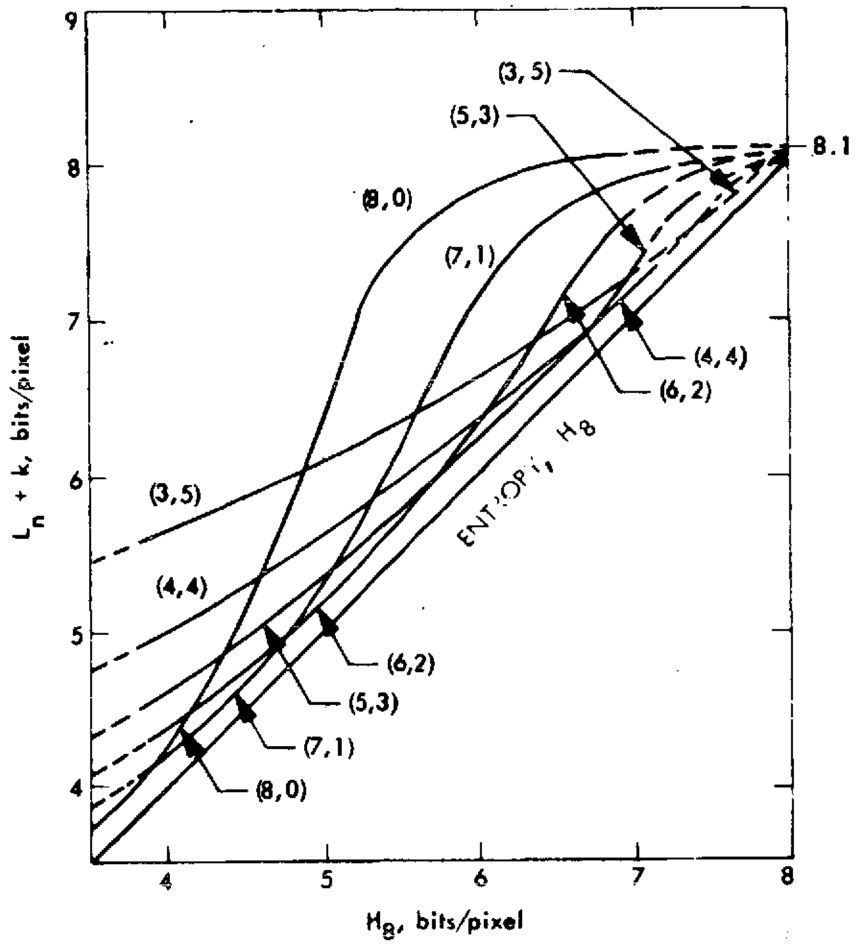


Figure 2.3-4. Split-Pixel Performance

As shown in Figure 2.3-4, not much compression is lost if the modes used is slightly wrong. As an example, if the block entropy is 6 bits/sample and the (6,2) mode is used instead of the optimal (5,3) mode, the data rate only increases by .15 bits/sample. Thus, not all possible modes need be provided in the compressor, simplifying both encoding and decoding operations. In the TRW simulation only three split-pixel modes are used. The entropy H of a segment of data is computed and the mode selected is determined as follows:

ENTROPY	MODE (n,k)
$H \leq 4$	(7,0)
$4 < H \leq 5$	(6,1)
$5 < H \leq 6$	(4,3)
$H > 6$	(3,4)

The (7,0) mode is not a split-pixel mode but the range of the basic compressor. Since the n most significant bits can be transmitted as FS, CFS, or $\overline{\text{CFS}}$, there are a total of twelve possible Rice modes allowed in the simulation.

2.3.3 Data Formatting

Since each block of data can be transmitted as FS, CFS, or $\overline{\text{CFS}}$ two block identification bits (BID) must be inserted before each block of compressed data. These two bits enable the decoder to know the mode used. Since each line will be encoded in one of four split-pixel modes, two line identification bits (LID) are inserted at the beginning of each new scan line. Following the two LID bits are the four intensity values which comprise the spectral components of the first element in the scan line. For split-pixel operation, following the two BID bits are the variable length Rice encoded n most significant data bits of the block. Following the Rice sequence are the k least significant bits of the block. The data format for the beginning of a scan line and the first (n,k) block is given below for a block of length L.

Several comments on the operation of program CRICE lend further insight into the techniques employed. First, the fundamental sequence (FS) is generated by the following steps which simulate the Rice wiggle operation:

(a) Have vector of 64 symbols stored in IRS.

(b) Set parameter IS to zero initially. On successive passes, IS=1, -1, 2, -2, etc.

(c) Generate the successive bits of fundamental sequence in IFS with a one bit for each entry of IRS where symbol level equals IS and a zero bit for each entry with a different symbol level.

(d) Keep track of location of symbols in IRS which produced a one bit in IFS. After each pass through IRS, reorder IRS by eliminating these symbol levels.

(e) Continue steps c and d, continually incrementing parameter IS until no symbols remain in IRS. At this point the fundamental sequence for the block is contained in vector IFS. If mode $\overline{\text{CFS}}$ is to be used, each entry in IFS is complemented.

Overhead bits are transmitted at the beginning of each new scan line with the following format:

		Initial Intensity Values				
Scan Line Identification	Mode SSDI/SSDIA	Band 1	Band 2	Band 3	Band 4	Mode of Split-Pixel
2 bits	1 bit	7 bits	7 bits	7 bits	7 bits	2 bits

New Scan Line Format

Depending on whether or not a split-pixel mode is being used for a block, two formats exist for the compressed data in a block.

Rice Mode	Compressed Block Bits
2 bits	? bits

Non Split-Pixel Block Format

Rice Mode	Compressed (n) block bits	(k) Residual Bits
2 bits	? bits	64 k bits

Split-pixel (n,k) Block Format

The initial two bits contained at the beginning of each block or line format is specified as follows:

00 → FS
01 → CFS
10 → $\overline{\mathrm{CFS}}$
11 → new scan line

2.3.4 The DCSTAT2 Simulation of the Rice Algorithm

The Rice simulation in DCSTAT2 given in section 2.5.2 computes the number of bits required for a Rice encoding of the transformed data. In order, to minimize the time required for the simulation, several shortcuts are used which introduce no error in the computation but are faster than the classical synthesis of the Rice code.

The entropy H of the input symbols is computed for each scan line and used to determine the split-pixel mode to use for the next scan line. The block size used is sixteen ground elements which implies 64 samples per block from all four spectra bands. For each block the FS is computed not by the Rice "wiggle" operation previously described but by recognition of the fact that the length of the FS (LFS) can be obtained by the algorithm:

$$\text{LFS} = \sum_{i=1}^{64} \left[2 \times \left|S_i\right| + \delta_i\right], \quad \delta_i = \begin{cases} 0 \text{ if } S_i > 0 \\ 1 \text{ if } S_i \leq 0 \end{cases}$$

Thus, for symbols 0, -2, 2, -1, 5 we obtain LFS = 1+5+4+3+10=23. The normalized length of FS is given by LFSN = LFS/64 and LFSN is used to compute the Rice mode to use for the most significant n bits of the data samples in that block. If LFSN $\leq$ 1.5 use $\text{C}\overline{\text{FS}}$, if 1.5 $\leq$ LFSN < 3.0 use FS, and if LFSN $\geq$ 3.0 use CFS.

If either CFS or $\text{C}\overline{\text{FS}}$ is used, the table given in Table 2.3-1 is used to determine the length of the coded sequence assigned to each block. Two more bits (BID) are added to each block total and a running total (NBRICE) of all bits required for encoding the samples in the scan line is maintained. At the end of each scan line this total is added to the thirty bits/scan line required for transmitting the LID and the first element values. Dividing this total by the number of samples in the scan line gives the average number of bits/sample requirec to Rice encode that line. In addition, DCSTAT2 computes the percentage of the time that the various Rice modes were used.

2.4 COMPUTER PROGRAMS

The computer programs used to simulate the basic compressions algorithms and to obtain the various statistical characterizations of the input data and the compressed data are discussed in this statistical measure section. The programs are flexible to permit the user selection of any or all of a number of different options. All programs have been written in FORTRAN IV or COMPASS and run on a CDC 6500 computer. Furthermore, to provide flexibility, the overall program was divided into several subprograms which can be used separately or in sequence. An overview of the data flow structure is provided by Figure 2.4-1. The interrelation and flow of the various functional programs is presented in Figures 2.4-2 through 2.4-5, including input and output data tapes.

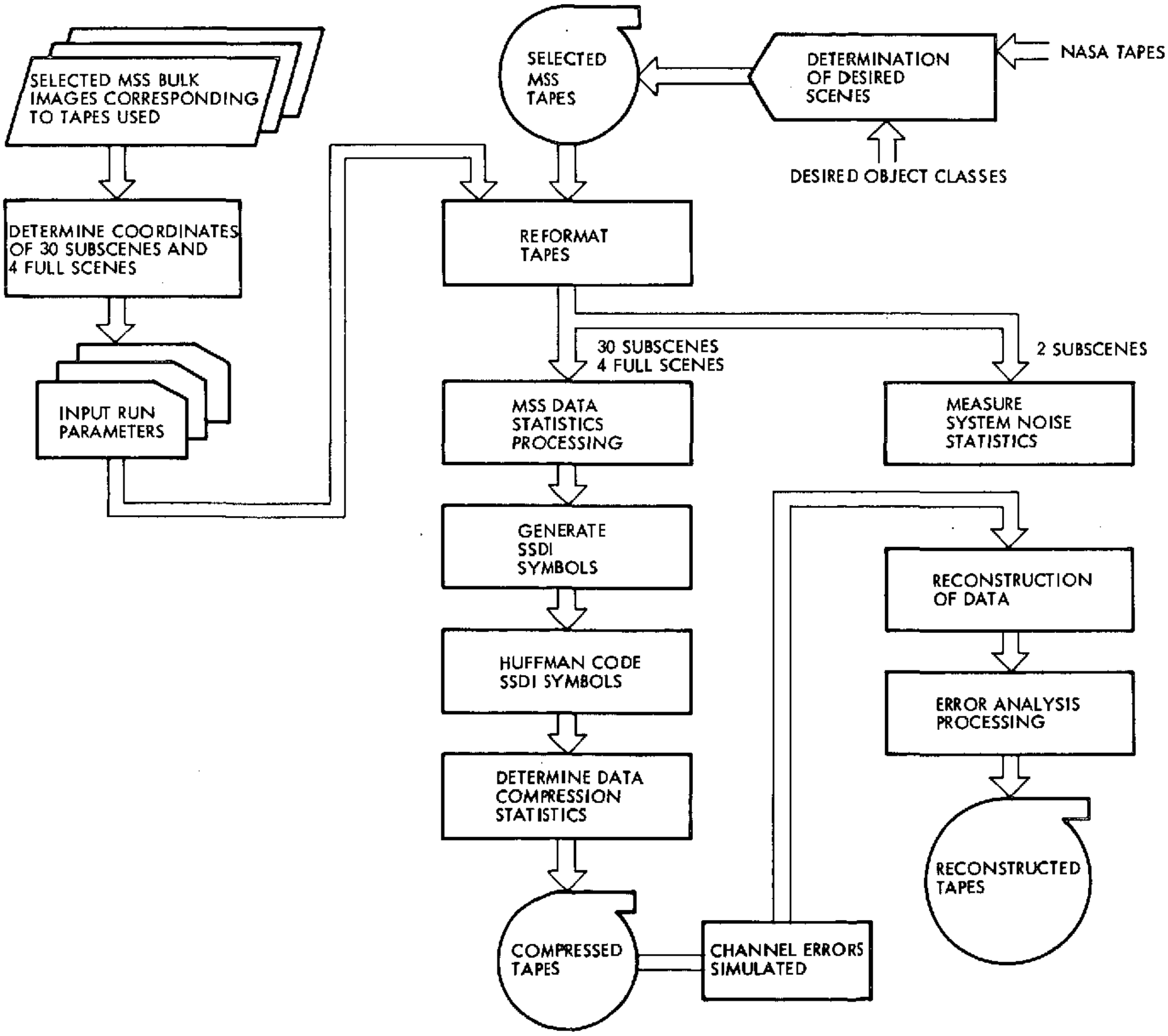

SELECTED MSS BULK IMAGES CORRESPONDING TO TAPES USED
SELECTED MSS TAPES
DETERMINATION OF DESIRED SCENES
NASA TAPES
DESIRED OBJECT CLASSES
DETERMINE COORDINATES OF 30 SUBSCENES AND 4 FULL SCENES
REFORMAT TAPES
INPUT RUN PARAMETERS
30 SUBSCENES 4 FULL SCENES
2 SUBSCENES
MSS DATA STATISTICS PROCESSING
MEASURE SYSTEM NOISE STATISTICS
GENERATE SSDI SYMBOLS
RECONSTRUCTION OF DATA
HUFFMAN CODE SSDI SYMBOLS
ERROR ANALYSIS PROCESSING
DETERMINE DATA COMPRESSION STATISTICS
RECONSTRUCTED TAPES
COMPRESSED TAPES
CHANNEL ERRORS SIMULATED

Figure 2.4-1. Overall Data Handling Flow Diagram

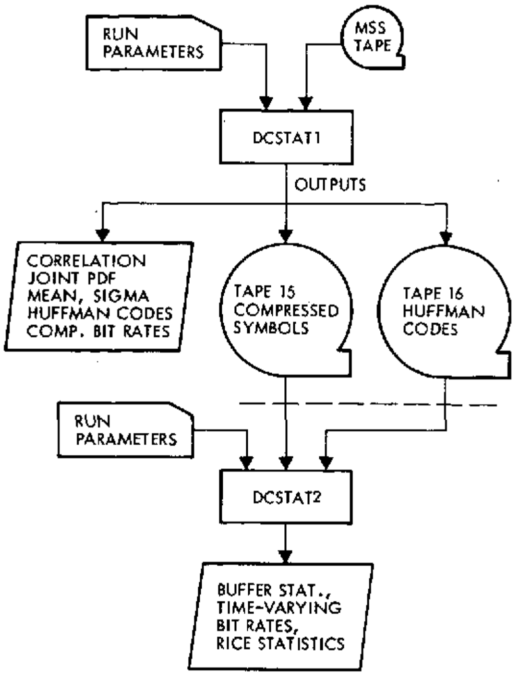

RUN PARAMETERS
MSS TAPE
DCSTAT1
OUTPUTS
CORRELATION JOINT PDF MEAN, SIGMA HUFFMAN CODES COMP. BIT RATES
TAPE 15 COMPRESSED SYMBOLS
TAPE 16 HUFFMAN CODES
RUN PARAMETERS
DCSTAT2
BUFFER STAT., TIME-VARYING BIT RATES, RICE STATISTICS

Figure 2.4-2. Data Flow Between DCSTAT1 and DSCTAT2

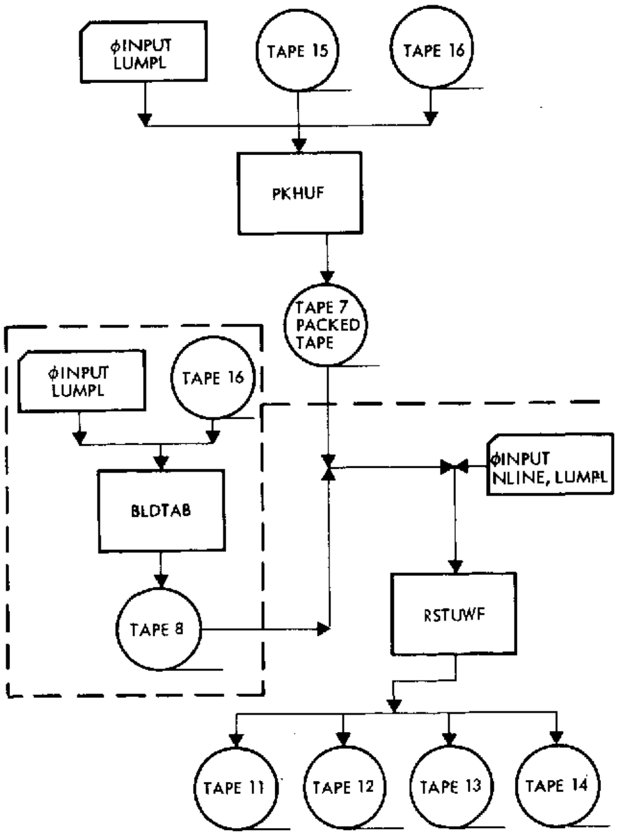


Figure 2.4-3. Huffman Compression and Reconstruction Flow Diagram

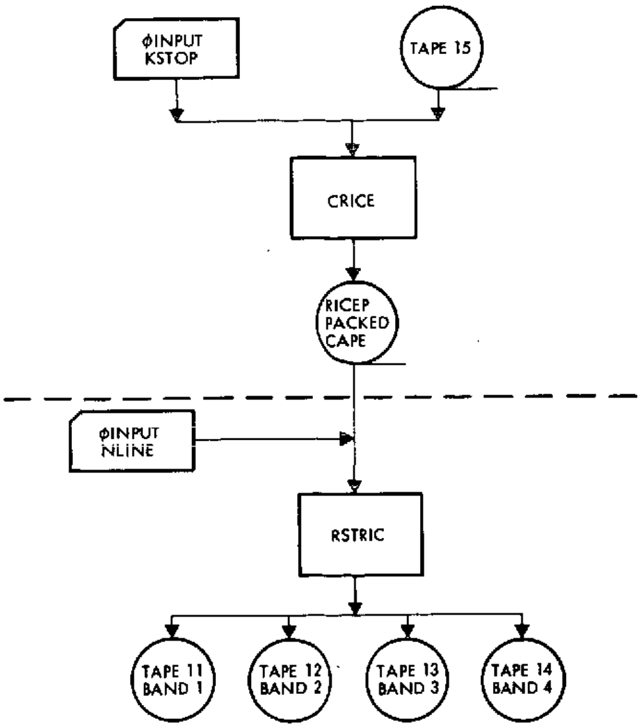


Figure 2.4-4. Rice Compression and Reconstruction Flow Diagram

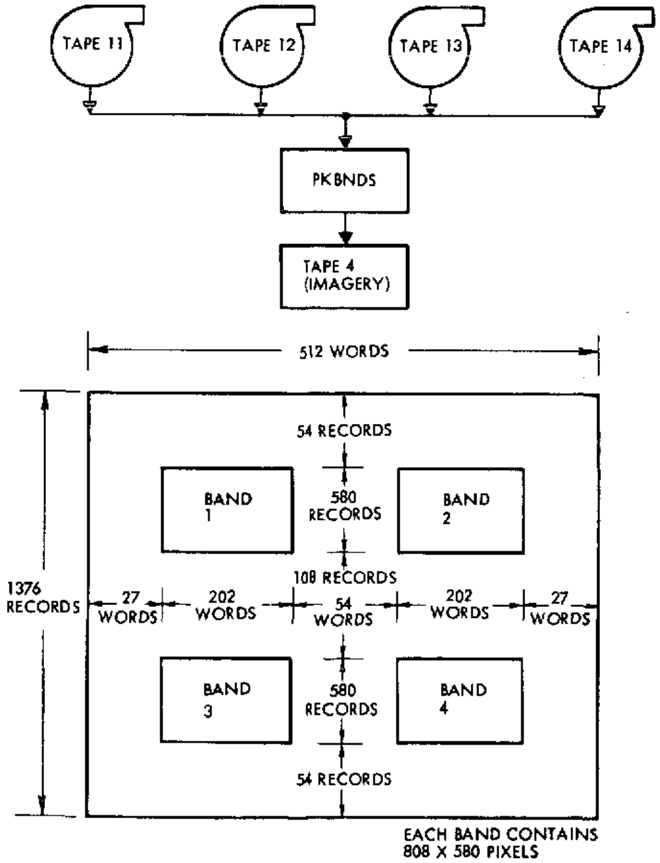


Figure 2.4-5. Generation and Format of TAPE 4 Containing Reconstructed Imagery

The program DCSTAT1 begins by extracting the data from the MSS tape and reformatting it. It then uses the appropriately formatted MSS data to compute scene statistics and compressed data statistics for the various compression algorithms selected. DCSTAT1 generates a tape containing the sequence of transformed differences obtained by the selected version of the SSDI , and generates the Huffmai code to be used on the scene.

DCSTAT2 uses the tape generated by DCSTAT1 and computes the time varying compression and buffer statistics resulting from the global Huffman, adaptive Huffman, and Rice encoding of the scene. DCSTAT2 also generates other characterizations of these data compression techniques for the selected scene. A two pass organization was required because the global Huffman code can be developed only after the probability density function has been measured for the entire scene.

For Huffman encoded data, program RSTHUF simulates the selected data compression technique by generating a compressed bit stream and reconstructing the data. This program operates on the tapes of difference symbols generated by DCSTAT1 Program BLDTAB generates the look-up table used by RSTHUF, the table being based on the Huffman code generated by DCSTAT1. Program PKHUF generates the Huffman coded data tape. The packed tape of a 7 track, 800 BPI, Fortran binary tape written in external format. The data is a continuous bit stream broken up into 288 60-bit word (2160 bytes) records.

Regardless of the coding algorithm used, the output of the reconstruction are four data tapes each containing one spectral band. These contain the reconstructed data for a scene comprising an area twenty-five nmi square. These four tapes are then packed to obtain TAPE 4 which contains the reconstructed data with spectral bands packed as shown in Figure 2.4-5.

The final step is the construction of the photographic copy of the imagery. TAPE 4 containing the images is processed on a General Dynamics L70 laser printer/scanner. The L70 is an eight-bit laser film writer with 256 intensity (grey) levels. The film writer constructs photographic negatives from which positive prints are then made at TRW. Note that each negative contains all four spectral bands, formatted as shown in Figure 2.4-5, to prevent negative and print processing variations which could otherwise occur. This technique also provides all spectral components of a scene on the same print to facilitate comparisons.

2.4.1 Program DCSTAT1

Program DCSTAT1 uses the MSS data tapes to generate the various statistical measurements of the input and SSDI-transformed data as well as tapes of the SSDI symbols. DCSTAT1 gives the operator flexibility in selecting options for each run. The output tape can be based onthe SSDI, SSDIA, or the SSDIAM algorithms.

The various statistical outputs which can be selected are:

- Data mean and variance in each spectral band and over all bands.
- First difference pdf in each spectral band for the SSDI, SSDIA, and SSDIAM transforms.
- Joint spectral-spatial correlation along the scan lines.
- Overall pdf for SSDI, SSDIA, SSDIAM, and Shell symbols.
- Huffman code for SSDI, SSDIA, SSDIAM, and Shell symbols.
- Scene entropy and average code length for SSDI, SSDIA, SSDIAM, and Shell transforms.

The flow of DCSTAT1 is given in Figure 2.4-6. The program is initialized and the various input parameter options are entered through namelist INPUT. These parameters (with default values indicated in parentheses) are:

- KSTOP - No. of scan lines in scene (100)
- JSTOP - No. of pixels in each scan line (100)
- BLUR - Blur option = "ØN" or "ØFF" (ØFF)
- JUMP - Correlation step sizes (1, 2, 4, 6, 8)
- BANDS - Band pairs used for probability ellipse (1, 2)
- CTAPE - Tape output = "ØN" or "ØFF" (ØN)
- IMODE - Tape output symbols = "SSDI", "SSDIA," or "SSDIAM," (SSDI)
- IPROB - pdf computation = "ØN" or "ØFF" (ØN)
- ICOR - correlation option = "ØN" or "ØFF" (ØN)
- IELPS - ellipse option = "ØN" or "ØFF" (ØN)
- ICOMP - compression option = "ØN" or "ØFF" (ØN)
- NSKIP - No. of initial records to skip
- ISLS - Starting scan line
- ISLE - Ending scan line
- IPS - Starting pixel location
- IPE - Ending pixel location

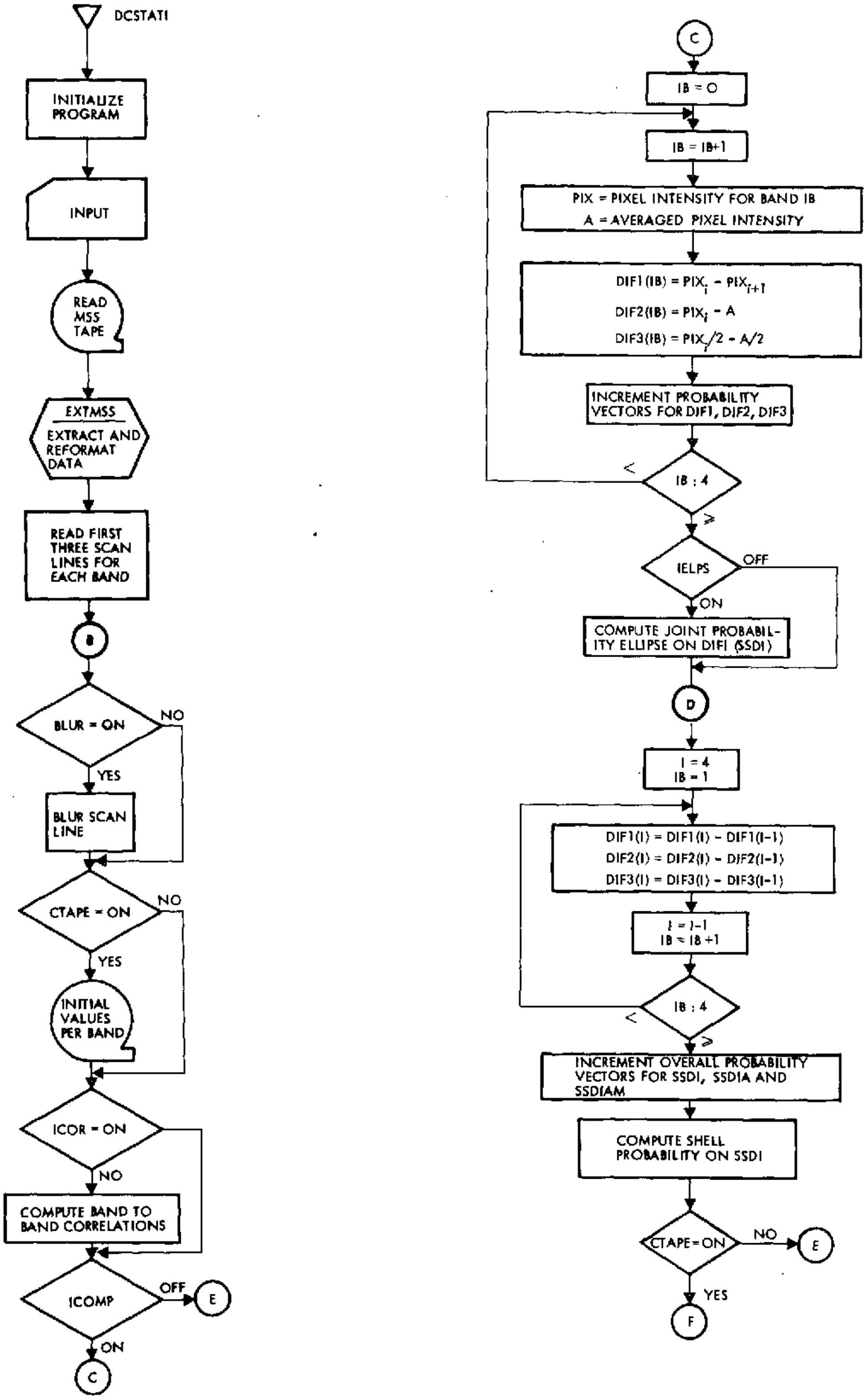

DCSTAT1
INITIALIZE PROGRAM
INPUT
READ MSS TAPE
EXTMSS
EXTRACT AND REFORMAT DATA
READ FIRST THREE SCAN LINES FOR EACH BAND
B
BLUR = ON
NO
YES
BLUR SCAN LINE
CTAPE = ON
NO
YES
INITIAL VALUES PER BAND
ICOR = ON
NO
COMPUTE BAND TO BAND CORRELATIONS
ICOMP
OFF
E
ON
C
C
IB = O
IB = IB+1
PIX = PIXEL INTENSITY FOR BAND IB
A = AVERAGED PIXEL INTENSITY
DIF1(IB) = $PIX_i - PIX_{i+1}$
DIF2(IB) = PIX_i - A
DIF3(IB) = PIX_i/2 - A/2
INCREMENT PROBABILITY VECTORS FOR DIF1, DIF2, DIF3
<
IB : 4
≥
IELPS
OFF
ON
COMPUTE JOINT PROBABILITY ELLIPSE ON DIFI (SSDI)
D
I = 4
IB = 1
DIF1(I) = DIF1(I) - DIF1(I-1)
DIF2(I) = DIF2(I) - DIF2(I-1)
DIF3(I) = DIF3(I) - DIF3(I-1)
I = I-1
IB = IB +1
<
IB : 4
≥
INCREMENT OVERALL PROBABILITY VECTORS FOR SSDI, SSDIA AND SSDIAM
COMPUTE SHELL PROBABILITY ON SSDI
CTAPE=ON
NO
E
YES
F

Figure 2.4-6. Flow of Program DCSTAT1

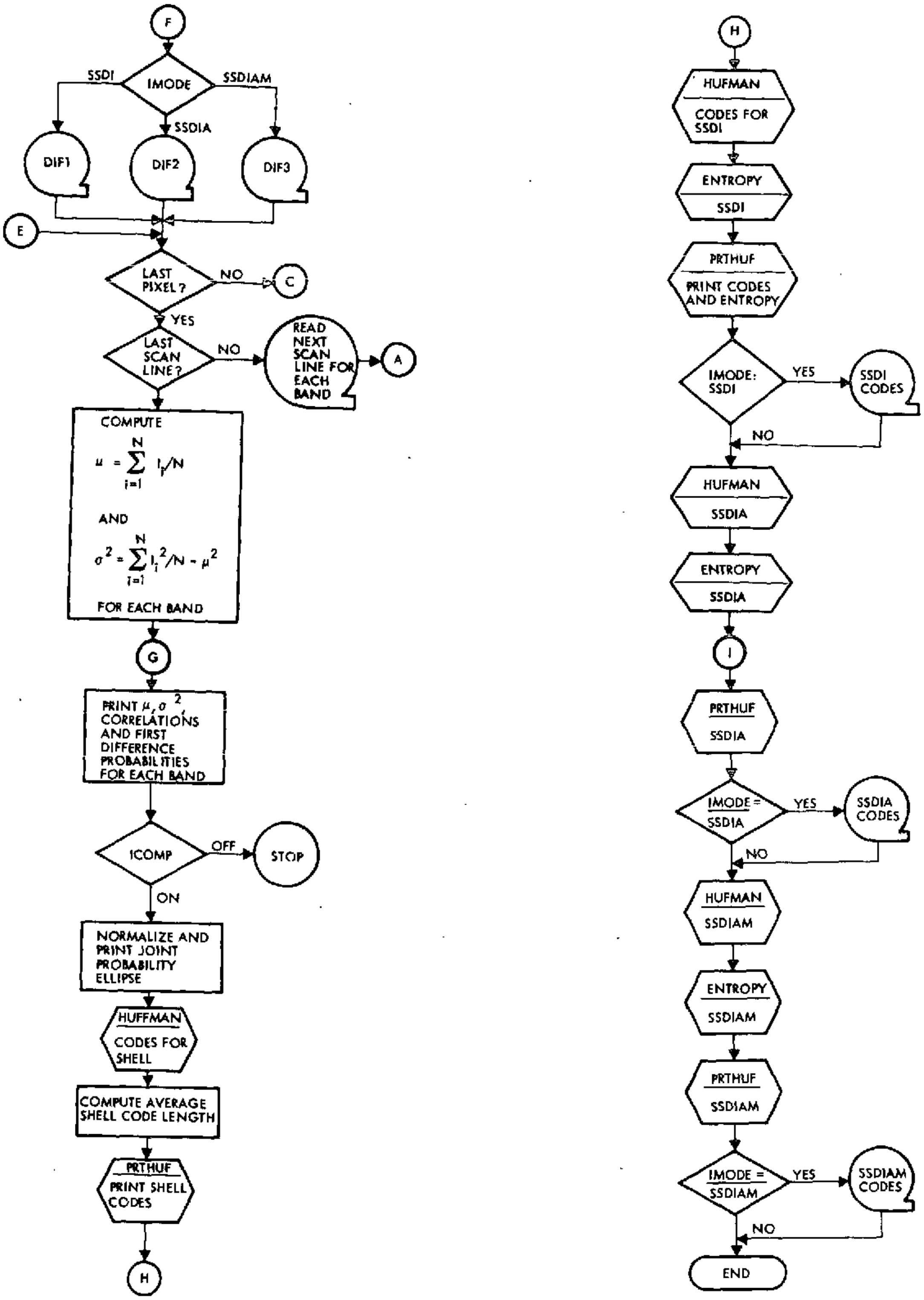


Figure 2.4-6. Flow of Program DCSTAT1 (Continued)

These last five parameters are used by subroutine EXTMSS which extracts and unpacks the MSS data from the input tape and reformats the data to a form acceptable by DCSTAT1. The flow of subroutine EXTMSS is given in Figure 2.4-7.

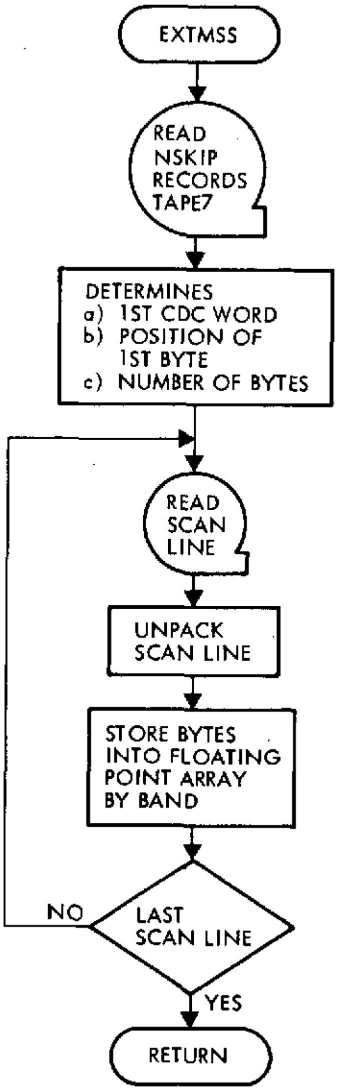


Figure 2.4-7. Flow of EXTMSS

The first three scan lines are read from the reformatted data for each spectral band. Three lines are initially read in each band to provide the information required if averaging and blur are to be performed. If BLUR is ON the inpu samples are blurred in each spectral band to simulate a decreased sensor resolutic using the equation:

$$I(i,j) = .68\ I(i,j) + .28\ [I(i,j-1) + I(i,j+1) + I(i+1,j) + I(i-1,j)]$$
$$+ .04[I(i-1,j-1) + I(i-1,j+1) + I(i+1,j-1) + I(i+1,j+1)]$$

Each intensity is used to update vectors S and S2 which will eventually contain the mean and second moment of the scene.

If CTAPE is ON, the initial intensities at the beginning of the scan line in each spectral band are written onto the output tape. If ICOR is ON the spectral-spatial correlation is performed by taking the normalized inner product of the vectors corresponding to pixels separated on a scan line by the distances given by the parameter JUMP. The correlation value for each pair of pixels is entered into array CØR. These values will later be averaged over the scene and printed as a correlation table.

The first differences are generated if ICOMP is ON. The first differences are obtained in three ways depending on whether SSDI, SSDIA, or SSDIAM is being simulated. For SSDI, the first differences are obtained by substracting successive pairs of intensities in each band, the difference being stored in array DIF1. For SSDIA, the first differences are obtained by subtracting each intensity from the average, A, of surrounding intensities, the differences being stored in array DIF2. For the SSDIAM, each intensity is mapped appropriately and then the first differences are obtained in the same fashion as used for the SSDIA, the result being stored in array DIF3. DIF1, DIF2, and DIF3 increment the probability vectors P1S, P1SA, and P1SAM. Simultaneously, array ELIPS is updated once per set of four spectral intensities for later use in generating the joint probability ellipsoid.

The second differences are then computed using the appropriate first order differences as computed by the SSDI, SSDIA, and SSDIAM. Three second differences and the appropriate first difference for spectral band one are used to increment the overall probability vectors P2S, P2SA, and P2SAM. These vectors are used to determine the Huffman codes for the scene. The differences symbols forming either the SSDI, SSDIA,or the SSDIAM are written on the output tape if CTAPE is ON. IMODE determines the compression mode written on the output tape.

The largest symbol difference in magnitude for each set of four spectral intensities is then used to determine which shell that set would occupy if Shell coding were used. Probability array PSH is then updated, corresponding to that shell. PSH is used to determine the optimum Huffman code for Shell coding

The above procedure is performed on a pixel by pixel basis. When the last pixel in a scan line is encountered the next scan line of data is read from the input tapes, stored, and the above computations are performed. After the last scan line has been read and all operations performed, computer output operations begin. First, parameter S is divided by the number of intensities used to generate the scene mean μ. Parameter S_2 is divided by the same number to obtain the second moment. The scene variance σ^2 is obtained by subtracting μ^2 from the second moment. These operations are performed for each spectral band and for all bands averaged together. The means, variances, and first difference probabilities are then printed.

The correlation is printed as a function of the step size for values of correlation between .71 and 1.00. The joint probability ellipse is printed for each pair of bands desired. In the printout the joint occurrence (0,0) is normalized to the value 100 and all other joint probabilities are normalized by the same factor to present an output product that can be easily interpreted. The SSDI, SSDIA, SSDIAM, and SHELL symbol probabilities are displayed in the ranoe [-18, 18] .

The remainder of the program generates the entropy and Huffman codes for the SSDI, SSDIA, SSDIAM, and SHELL encoding modes. All four calls to subroutine Huffman are alike, the only difference being the probability vector used to compute the appropriate Huffman code. Given the probability of occurrence of each symbol over the scene, Huffman returns the sequence of coded bits associated with each symbol and the average code length. Subroutine Huffman is fully described in section 2.4.5. Scene entropy is computed for each coding made from the same probability vector by the equation:

$$H = -\sum_{i} p(i) \log_2 p(i)$$

Depending on the IMODE specified by the operator, either the SSDI, SSDIA, or SSDIAM Huffman code table is written onto TAPE16 for use by DCSTAT2 or BLDTAB.

After specifying the various input parameters to DCSTAT1 operation is automatic and no further operator interaction is required so that the program can be submitted either through a terminal or by batch processing. If all options are ON program DCSTAT1 requires about 1.2 milliseconds per intensity samole.

2.4.2 Program DCSTAT2

Program DCSTAT2 accepts the tapes containing the SSDI,. SSDIA, or SSDIAM symbols and the Huffman code computed by DCSTAT1 for that scene and computes the time-varying statistics for the global Huffman, and Rice encoding. The flow of DCSTAT2 is given in Figure 2.4-8.

Input parameters are accepted via namelist INDC2. These parameters are IBUF1, IBUF2, MODE, and PMAX. IBUF1 and IBUF2 denote the buffer output rate in bits/sample, with default values 3.0 and 3.5. Parameter MODE determines the outputs desired and has three values:

MODE = 1 → Buffer statistics and global Huffman statistics

2 → Rice and adaptive Huffman statistics

3 → All of the above (default value)

Parameter PMAX permits varying the total probability of these symbols included in the lumped grouping for the generation of the adaptive Huffman code.

Initially, DCSTAT2 reads input tape TAPE16 which contains the code table, of length 256, associating each input symbol read from TAPE15 to the global Huffman code as generated in DCSTAT1.

The first scan line of symbols is read from TAPE15 by COMPASS program LININ. If MODE is set to either 2 or 3, HPAH is called to compute the probability distribution of the transform symbols for that scan line. At the end of the scan line this distribution is used by subroutine HUFMAN to develop the Huffman code which will be used to encode the symbols from the following scan line. This technique is used for all scan lines in the adaptive Huffman mode except for the first line, since no a priori information is available there. The first line is encoded a posteriori by the Huffman code developed for that line. Thus, the first two scan lines of data have the same code. In addition, the entropy of the scan line symbols is computed by HPAH for use in Rice encoding. The normal call to LININ, reading the next scan line, follows B. This read is skipped for the first scan line since it has already been read to initiate the adaptive Huffman mode. If an end-of-file is encountered by LININ, the program goes to H to print the overall compression achieved on the data.

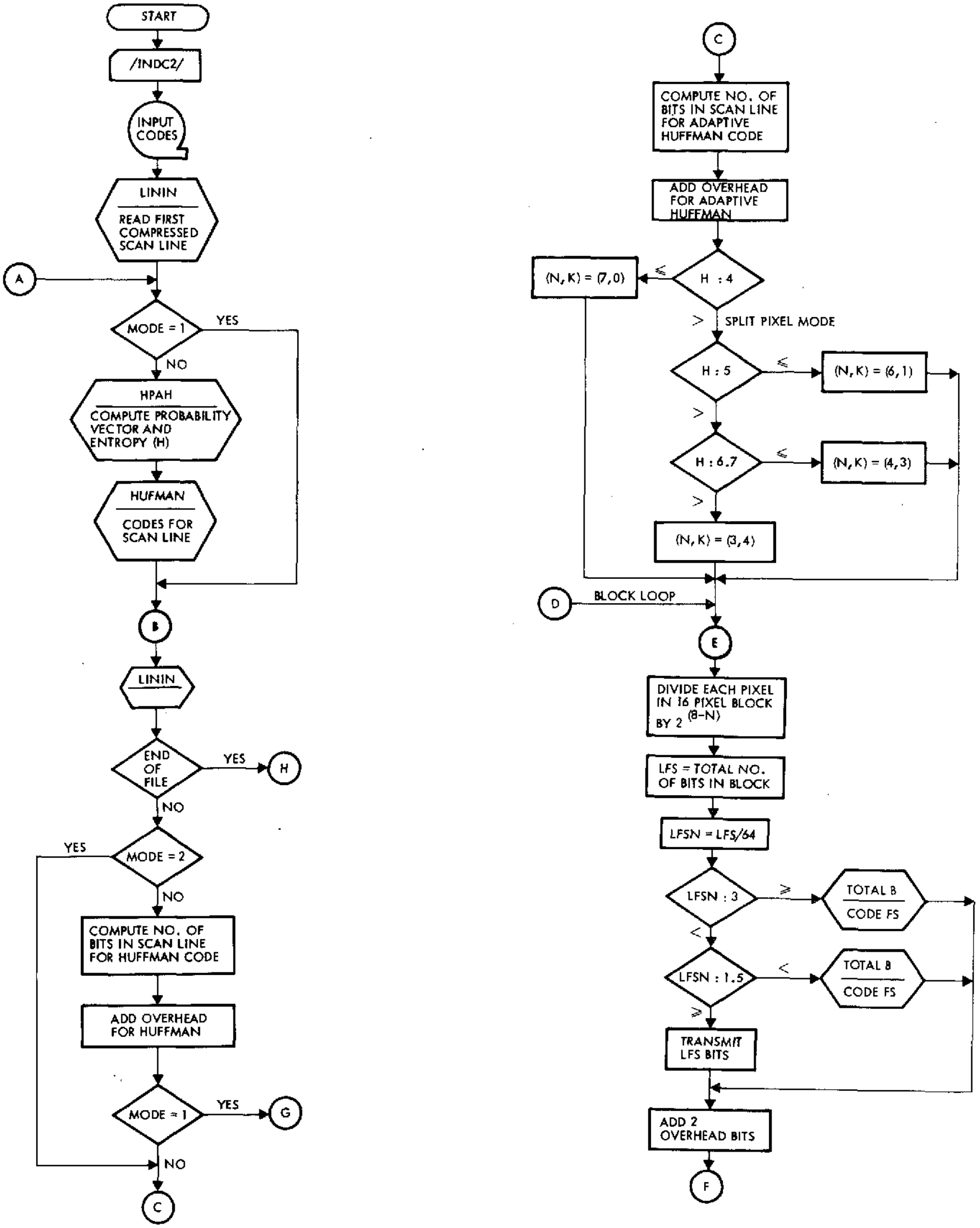


Figure 2.4-8. Flow of Program DCSTAT2

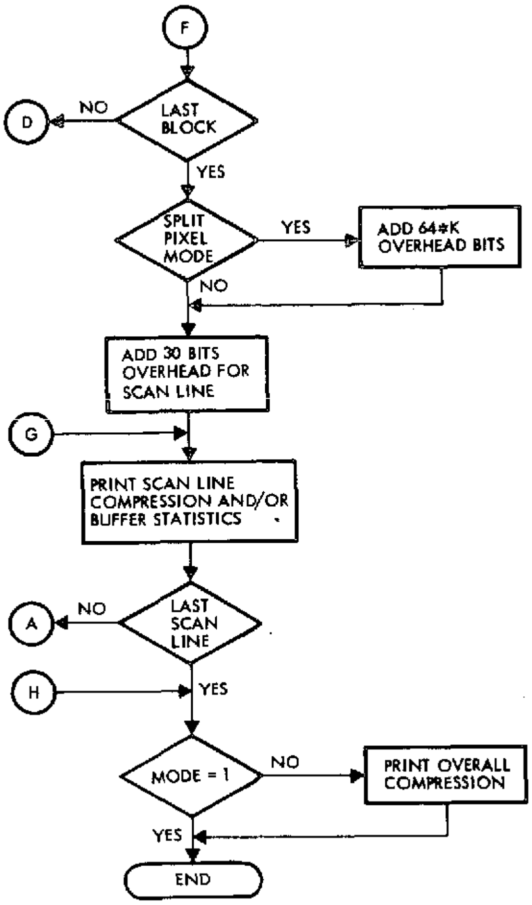

F
LAST BLOCK
NO
D
YES
SPLIT PIXEL MODE
YES
ADD 64*K OVERHEAD BITS
NO
ADD 30 BITS OVERHEAD FOR SCAN LINE
G
PRINT SCAN LINE COMPRESSION AND/OR BUFFER STATISTICS
LAST SCAN LINE
NO
A
H
YES
MODE = 1
NO
PRINT OVERALL COMPRESSION
YES
END

Figure 2.4-8. Flow of Program DCSTAT2 (Continued)

If no end-of-file is encountered a check is made for MODE=2. If MODE is not 2 the number of bits required for encoding that line by the global Huffman code is determined using the stored table. Each transform symbol is used to call the code table and the number of bits required to code that symbol is returned. After all symbols from that scan line have been converted to bits, the total number of bits obtained is added to the number of overhead bits required and this sum is divided by the total number of symbols encoded to obtain the average number of bits required for that line of data. For the global Huffman the overhead per scan line is equal to the number of bits required to denote the beginning of a new scan line and the number of bits used to send the values of the first intensities in the scan line. The first pixel intensities in the line are transmitted directly and not by transform symbols in order to prevent propagation of possible errors from one scan line to the next line. The format used in DCSTAT2 for denoting the beginning of a scan line is the same as that given in the appendix on ground processing of images. The prefix code is transmitted followed by a string of eight zero bits to initiate the scan line and the next 28 bits give the four 7-bit intensity values of the first elements in each spectral band.

If MODE=1, the flow is transferred to G where the compression is printed for global Huffman encoding as the average number of bits per pixel required for that line. The average number of bits per pixel out of the buffer (IBUF) is subtracted from the average no. of bits per pixel required for compression to yield the average change in the buffer for that line. This amount is added to the buffer contents left from the previous line to yield the total buffer fill. If IBUF is greater than the average number of bits put into the buffer, underflow occurs and a 0 is printed for the buffer statistics of that line. In general, IBUF would initially be set at about the average bits/pixel required for the scene and variations of buffer fullness would be observed. After a run of DCSTAT2, buffer statistics may reveal runs of underflow for segments of the scene where data activity is much less than the scene average. Also, the more serious problem of overflow may result due to areas of the scene where data activity is much higher than the scene average. In such cases DCSTAT2 can be rerun with an appropriate change in IBUF value.

If MODE is not equal to 1, the average bit rate for that line is computed for adaptive Huffman coding. The Huffman code developed and stored in table look-up form for the previous scan line is used to convert input symbols into bits in a fashion analogous to that used for global Huffman encoding. The overhead required for adaptive Huffman coding is also identical to that required for global Huffman coding, namely the number of bits required for denoting the start of a new scan line and the 28 bits giving the first four intensity values.

The first operation performed for Rice encoding is the determination of which split-pixel mode to use on the line, if any. Several checks of the previously computed line entropy H are made. If $H \leq 4$, no split-pixel mode is used ($n=7$, $k=0$). If $4 < H \leq 5$ the ($n=6$, $k=1$) mode is used. If $5 < H \leq 6$, the ($n=4$, $k=3$) mode is used. If $H > 6$, the (3, 4) mode is used. As given in section 2.1.6 the n most significant bits are Rice encoded and the k least significant bits are transmitted directly to provide strictly information preserving data compression. If distortion is desired, k can be set in the program to a value less than 7-n.

The Rice coding is performed on blocks of data in the scan line. This block length is preset to 16 elements along the line but can readily be changed to another value if desired. The total number of bits in the block fundamental sequence (LFS) is first computed by determining the number of bits required for each symbol in the block. The algorithm used is described in section 2.3. The normalized fundamental sequence length (LFSN) is obtained by dividing LFS by the total number of symbols in the block. Length LFSN is tested to determine whether the fundamental sequence (FS) should be transmitted directly, whether the fundamental sequence should be coded (CFS), or whether the fundamental sequence should first be complemented and then coded (C$\overline{F}$S). If LFS is between 1.5 and 3 bits/symbol FS is transmitted, if LFS $\geq$ 3 bits/symbol CFS is transmitted, and if LFS < 1.5 bits/symbol C$\overline{F}$S is transmitted. Subroutine TOTALB assigns to FS or $\overline{FS}$ the code described in Appendix C and computes the total number of bits required for coding that block. If FS is transmitted directly, the number of bits required for that block is LFS. Two overhead bits are added per block to denote the Rice mode used for coding that block of data.

After the last block of data in the scan line has been encoded,additional overhead is added to the total number of bits already computed. If a split-pixel mode was used for that line, overhead is added for transmitting the k least significant bits. This overhead amounts to k times the number of symbols in the line of data. At the beginning of each scan line overhead is used to give the decoding algorithm line initialization information including the 7-bit intensities of the first element in each spectral band.

At the end of each line the compression statistics are printed both on a line-by-line basis and globally. Depending on the value assigned to MODE, either the buffer statistics and the average bits/sample for the global Huffman coding, the average bits/sample for adaptive Huffman and Rice coding, or all four are printed. After the last scan line of data, the percentage occurrence of the various Rice modes are printed (if MODE is not set to 1).

2.4.3 Program BLDTAB

Program BLDTAB generates the look-up decoding table used by program RSTRCT to reconstruct the compressed data. BLDTAB initially reads TAPE16, the tape generated by DCSTAT1 which contains the Huffman code developed for the given scene. Other inputs entered for each run are N, the number of code words, LCØDE, the lumped (grouped) prefix code word, and LUMPL, the number of bits in the lumped code word array. IHC contains the Huffman code words from TAPE16 and array IC gives the corresponding number of bits in each code word.

The program generates a table ITAB of length 2^{12} corresponding to all possible sequences of twelve bits. Each entry in the table contains twelve bits of information, the most significant eight bits giving the first difference symbol which is Huffman decodable in the twelve bit address of that entry and the least significant four bits give the number of bits contained in that decodable word. Both pieces of data are required by program RSTRCT. The structure of BLDTAB is given in Figure 2.4-9.

If a non-grouped symbol Δ_i is present which produces a code word of length j bits, the value (Δ_i, j) is entered at all locations in the table which have the binary representations with the most significant j bits equal to the code word C_i. The number of entries of (Δ_i,j) is equal to 2^{12-j}. For entries where the most significant bits are the lumped code word C_L (the length of C_L is normally constrained to be at most four bits), the following eight bits give the symbol level which occurred or, if the eight bits are all zeros, denote the start of a new scan line. The entries in the table corresponding to addresses beginning with codeword C_L use the following eight bits in the address as the actual level of the grouped symbol which lies in the range [-128,128]. Parameter IT shifts the lumped code word to head the 12 bit string. ISFT equals the total number of bits in C_L and the eight bits following C_L. IS is the number of bits left in the twelve bit string in the event that C_L occurs (IS =12-ISFT).

The loop ending at B sets up all lumped entries in the table required by the given codeword. If IS is zero, only 256 entries need be set since ISFT = 12. If IS is greater than zero, each of the 256 lumped values must be put into 2^{IS} locations. ID, initially zero, is shifted IS bits and IW represents the appropriate

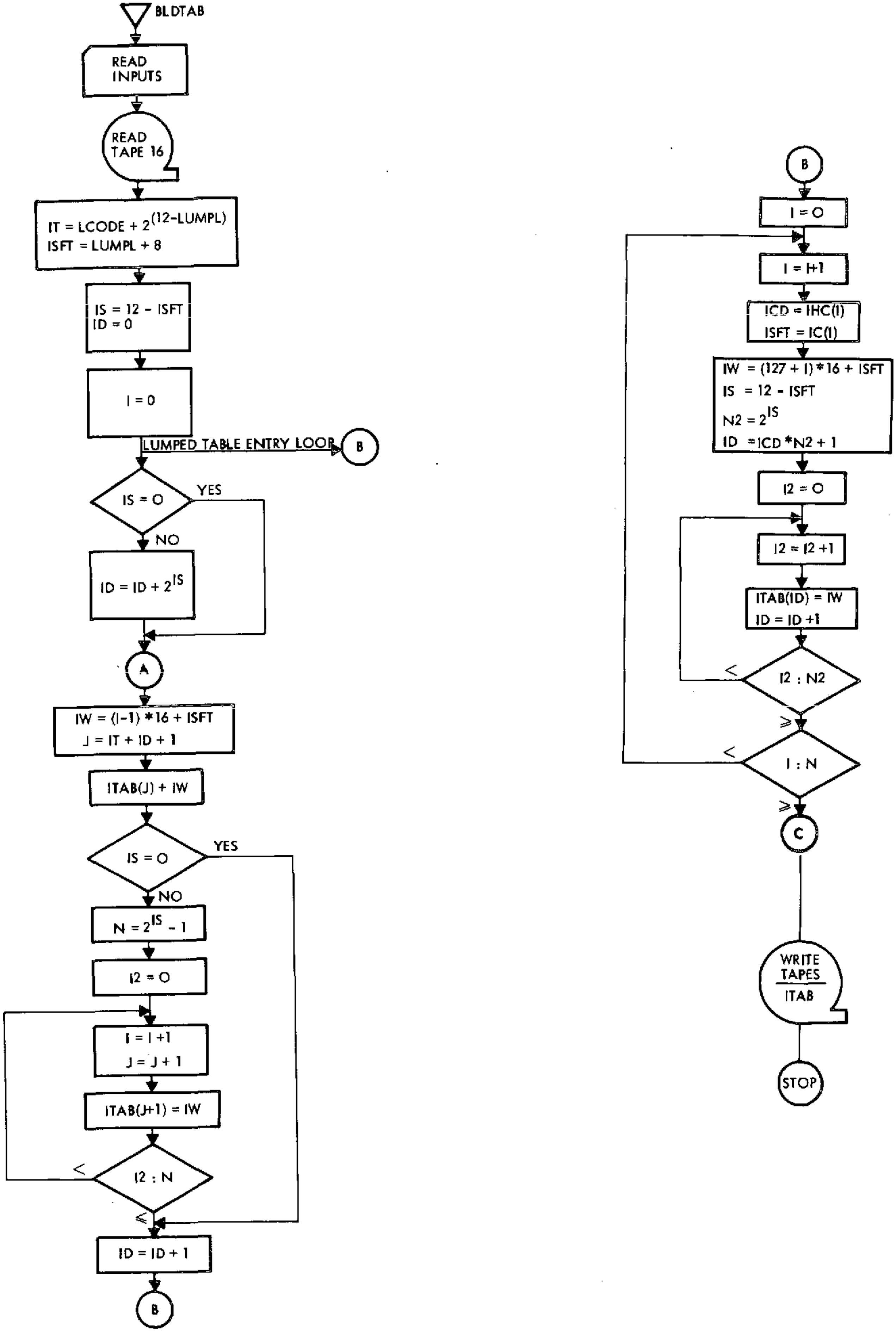


Figure 2.4-9. Flow of Program BLDTAB

table entry of level plus bits to be stored in ITAB at address J. As mentioned, if IS is not zero, 2^{IS} -1 other locations must be filled with the same information. After ID is incremented by one, the loop is continued.

Following generation of the lumped symbol entries, the non-lumped symbols are stored. In each loop, ending at C ICD represents the Huffman code word and ISFT represents the number of bits in that word. IW represents the table entry (Δ_i,j) to be stored in 2^{N2} locations of ITAB. These entries are set by loop C.

After all of the non-lumped symbols have been set, the entire 2^{12} entries of ITAB have been generated. Figure 2.4-10 gives a segment of table ITAB. The top section of Figure 2.4-10 shows a portion of the lumped symbol table entries and the remainder gives entries corresponding to symbol levels -2 and 2. Table ITAB is written onto TAPE8.

2.4.4 Program PKHUF

Program PKHUF, written in COMPASS, generates the compressed data tape using the Huffman code from TAPE16 and the symbol tape, TAPE15. PKHUF reads the successive transform symbols from TAPE15 and looks up the corresponding code word from the stored table. These code words are then packed onto the output tape, TAPE7, using subroutine PACODE. The code bits are packed in a manner permitting a code word to overlap computer words or tape records. PKHUF is shown in Figure 2.4-11.

At the beginning of a scan line, the lumped code bits C_L are transmitted followed by eight zeros. This is followed by a single bit denoting the encoding mode used for the scan line. Provision is made for either SSDI or SSDIA encoding and this first bit informs the reconstruction program RSTRCT as to the mode used. The following 28 bits represent the actual seven bit intensities of the first scan line element in each of the four spectral bands. This information serves to initiate the algorithm used for reconstruction and prevents the propagation of possible errors from one scan line to the next.

Subroutine INLINE reads in each new scan line of data. For this new scan line, PACODE packs code word C_L and the one bit of mode data onto the output tape TAPE7. The following four words read from TAPE15 contain the first intensiti from each spectral band. These 28 bits are packed onto the output tape. For the

```
 17,12    18,12    19,12    20,12    21,12    22,12    23,12    24,12    25,12    26,12
 27,12    28,12    29,12    30,12    31,12    32,12    33,12    34,12    35,12    36,12
 37,12    38,12    39,12    40,12    41,12    42,12    43,12    44,12    45,12    46,12
 47,12    48,12    49,12    50,12    51,12    52,12    53,12    54,12    55,12    56,12
 57,12    58,12    59,12    60,12    61,12    62,12    63,12    64,12    65,12    66,12
 67,12    68,12    69,12    70,12    71,12    72,12    73,12    74,12    75,12    76,12
 77,12    78,12    79,12    80,12    81,12    82,12    83,12    84,12    85,12    86,12
 87,12    88,12    89,12    90,12    91,12    92,12    93,12    94,12    95,12    96,12
 97,12    98,12    99,12   100,12   101,12   102,12   103,12   104,12   105,12   106,12
107,12   108,12   109,12   110,12   111,12   112,12   113,12   114,12   115,12   116,12
117,12   118,12   119,12   120,12   121,12   122,12   123,12   124,12   125,12   126,12
127,12   128,12    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4
 -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4    -2, 4     2, 4     2, 4
  2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4
  2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4
  2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4
  2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4
  2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4     2, 4
```

Figure 2.4-10. A Segment of Table ITAB

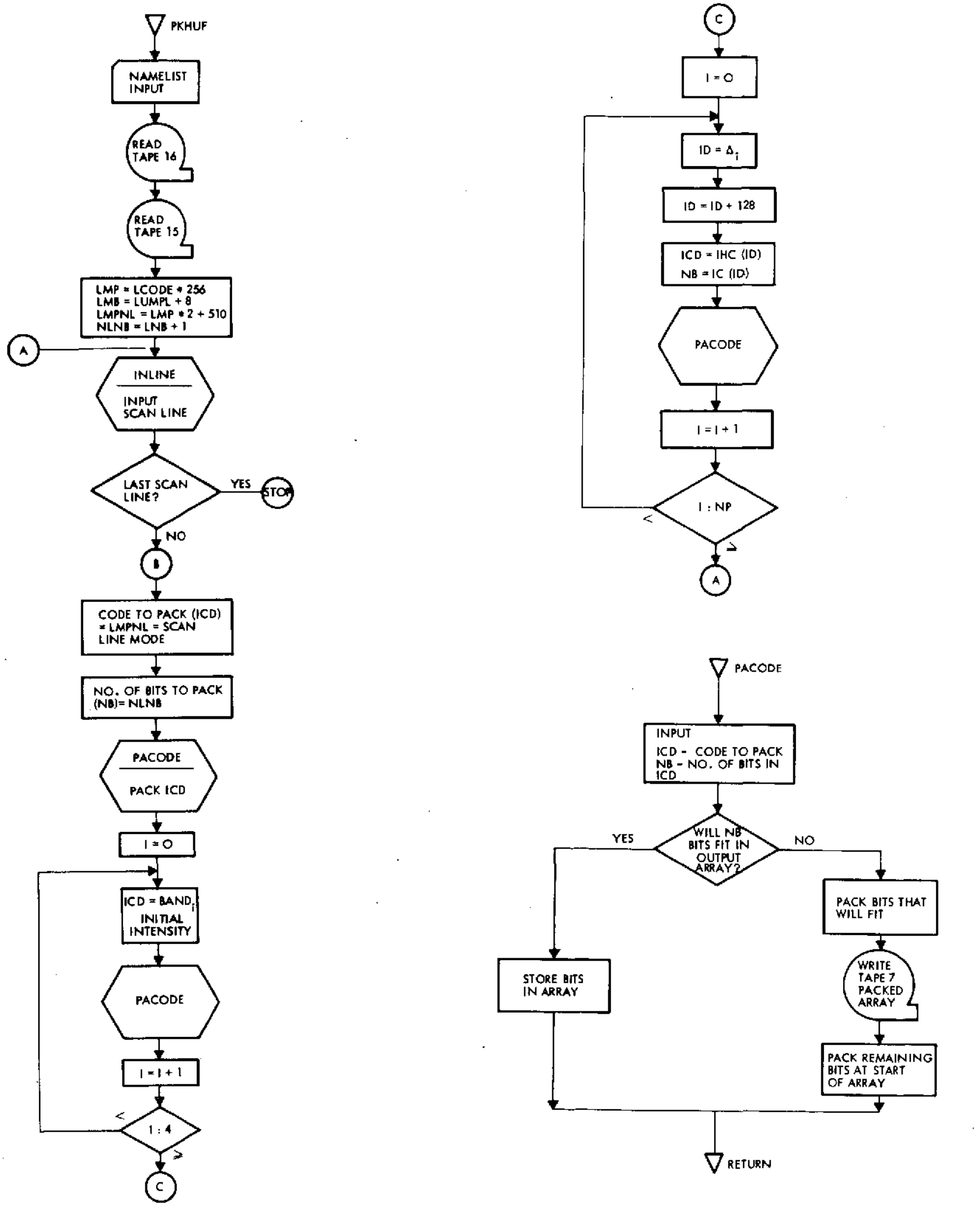


Figure 2.4-11. Flow of Program PKHUF

rest of the scan line, the transform symbols Δ_i from TAPE15 are used to obtain the associated code words C_i which are then packed onto TAPE15. There are NP symbols per scan line, each stored as a level between -128 and +128. To access code word table IHC, 128 is added to each symbol level read from TAPE15.

The program proceeds from one scan line to the next, packing the bits as shown into one long bit stream until the last scan line is encountered at which time the program stops and an end of file is generated.

2.4.5 Program RSTHUF

Program RSTHUF reconstructs the data using the compressed data on TAPE7 and the look-up decoding table ITAB from TAPE8. To initialize RSTHUF, table ITAB is read from TAPE8 and input parameters NLINE and LUMPL are entered. NLINE is the number of lines to be reconstructed and LUMPL is the number of bits in the lumped prefix code word. Masks MSK1, MSK2, and MSK3 are set to mask off the seven bit intensity values from the compressed data tape (MSK3) and to separate the eight bit symbol level (MSK1) and the four bit shift value (MSK2) from each twelve bit entry in table ITAB (see Figure 2.4-12).

Parameters IL, the scan line designator, is initialized to zero and ISFT, the number of shifts of the input data, is set to twelve and the first coded record is read via subroutine INTAP. Loop 100 reconstructs each set of four intensities corresponding to a ground picture element. Loop 100 obtains the set of four transform differences to be decoded by use of subroutine GET12. GET12 returns each twelve bit sequence from the input record using the previous shift value ISFT and stores this word in IWD. IWD is used as address to table ITAB, returning the entry ITAB (IWD). This entry is masked by MSK2 to return ISFT, the number of shift required for the next decoding operation, and ID(I) the first symbol in IWD. The value ID(I) is then shifted left four places to obtain the true value of the symbol.

ID(I) is compared to the value 256 in order to check for the occurrence of a new scan line. The start of a new scan line is denoted in the compressed data by the lumped prefix code word followed by eight zeros. The corresponding ITAB entry gives the symbol value 256 and ISFT = 12.

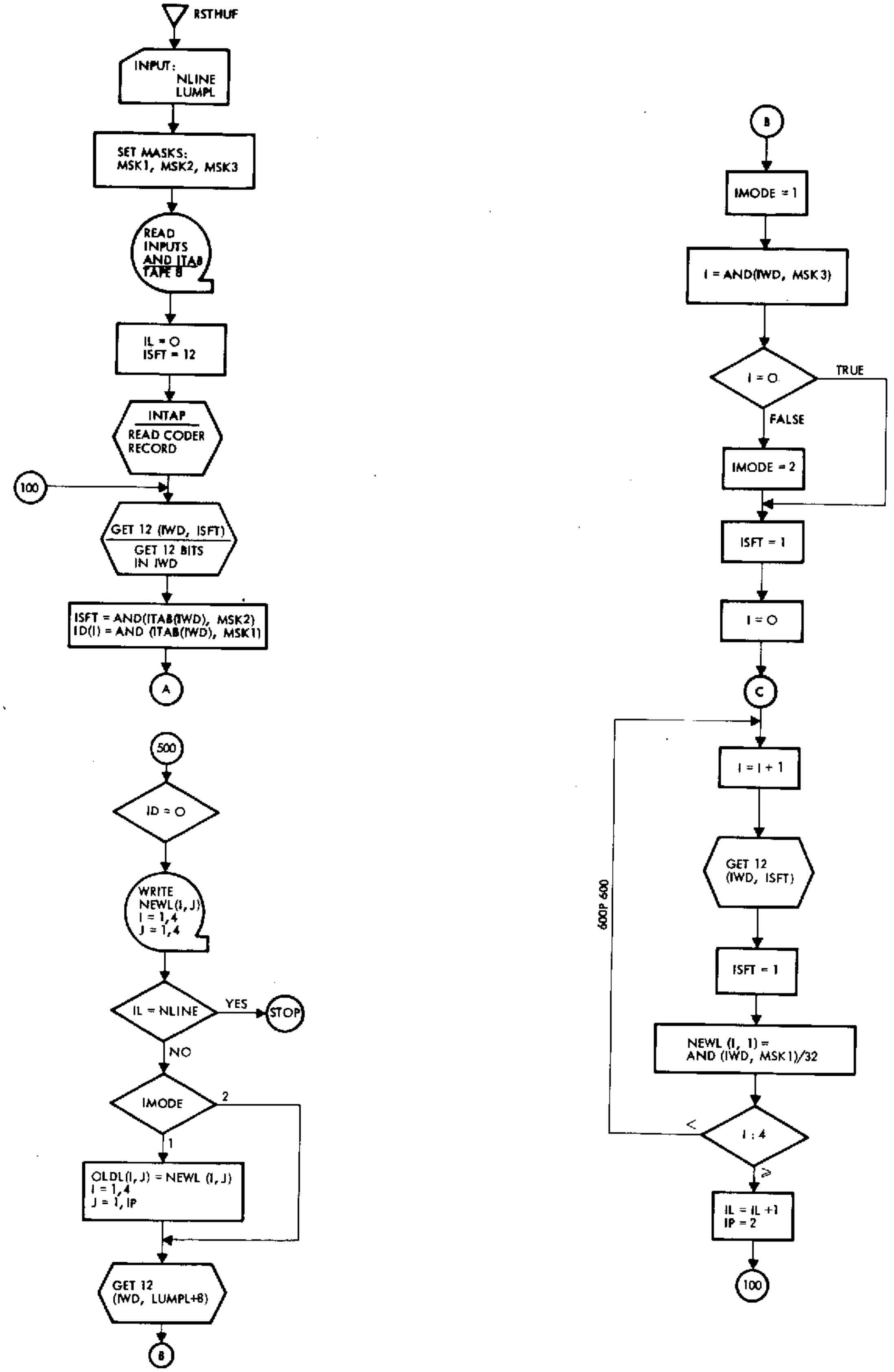


Figure 2.4-12. Flow of Program RSTHUF

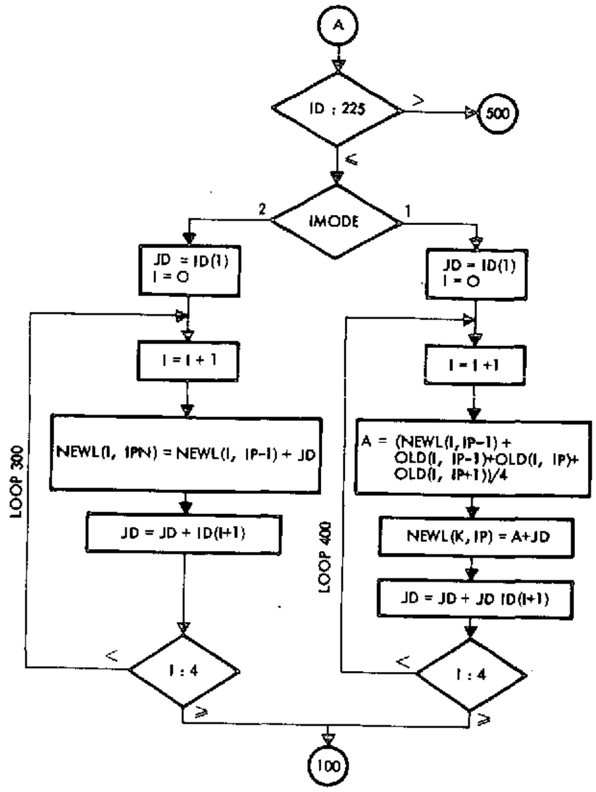


Figure 2.4-12. Flow of Program RSTHUF (Continued)

At the start of a scan line flow shifts to point 500. If not the first scan line, the previous line of reconstructed data (NEWL) is written onto the output tape. If not the last scan line, subroutine GET12 shifts the input data to get the next twelve bits of code. The first bit gives the mode (SSDI or SSDIA) used for that scan line. That bit, masked by MSK3, is checked to set IMODE. If I = 0, IOMCE = 1 for SSDIA, otherwise IMODE = 2 for SSDI. The following 32 bits are the true intensity values of the first element in each of the four spectral band. Loop 600 reconstructs these intensities in NEWL(I,1). After regenerating these four intensities, line counter IL is incremented by one and pixel counter IP is set to 2.

Normal SSDI or SSDIA reconstructedis performed if ID(I) $\leq$ 255. IMODE determines the algorithm used on that line of data. If IMODE = 2, SSDI reconstruction is performed by loop 300 using the set of four symbol values obtained in loop 100 and the previously reconstructed pixels in the line. These reconstructed intensities are stored in NEWL (I, IP), where I denotes the spectral band and IP denotes the position of the element along the scan line.

If IMODE = 1, SSDIA reconstruction is performed by loop 400 for each set of four symbols obtained by loop 100. First, as described on page an average of the four appropriate reconstructed intensities is computed in each spectral band. These averages (A) are used together with the difference symbol values to reconstruct the intensities in NEWL. After the last line has been reconstructed and written on the output tape, IL = NLINE and RSTRCT terminates.

The flow of subroutine GET12 is given in Figure 2.4-13. Subroutine GET12 extracts each twelve bit sequence from the input tape, storing the sequence in IWD. Parameter ISFT is entered through the call and gives the number of shifts required to correctly position the new bits in IWD so that a new code begins in the first bit position of IWD. GET12 handles the various bookkeeping and bit picking tasks required when code words overlap successive computer words or tape records.

2.4.6 Program CRICE

Program CRICE generates the Rice-compressed data tape RICEP. CRICE reads TAPE15, generated by DCSTATI, which contains the sequence of difference symbols (either SSDI or SSDIA), Rice-encodes these symbols, and outputs the data into a continuous packed bit format onto tape RICEP. The flow of CRICE is given in Figure 2.4-14 and subroutine SPLIT, called for the split-pixel mode, is detailed in Figure 2.4-15. Parameter KSTOP specifies the number of scan lines to be compressed. No input specifying the number of pixels per scan line is required since information designating the start of a new scan line is contained in TAPE15.

For each scan line, the program begins by reading in a line of data symbols and computing the probability distribution function of the symbols for subsequent use in generating the line entropy. The decision as to the split-pixel mode to be used on a line of data is based on the symbol entropy of the previous line. For this reason, the first scan line of data is encoded with the split-pixel mode OFF. With subsequent lines for which a split-pixel mode is required, subroutine SPLIT is called. For a given (n,k) mode each integer symbol Δ_i is divided by 2^k to obtain the new symbol values to be Rice encoded. The difference between this scaled symbol and Δ_i, IRR, is generated for subsequent direct transmission. This operation is described in section 2.3.2.

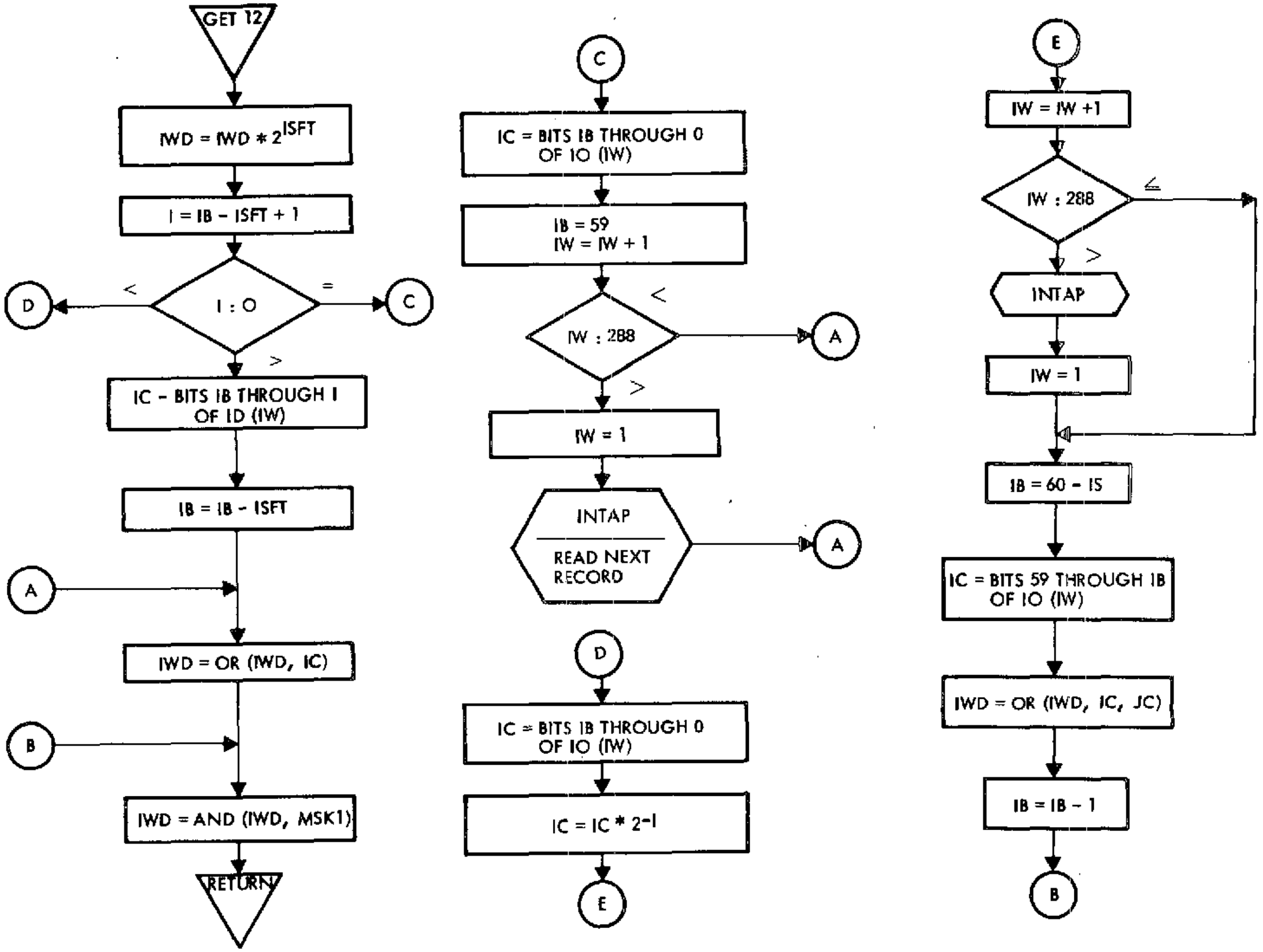


Figure 2.4-13. Flow of Subroutine GET12

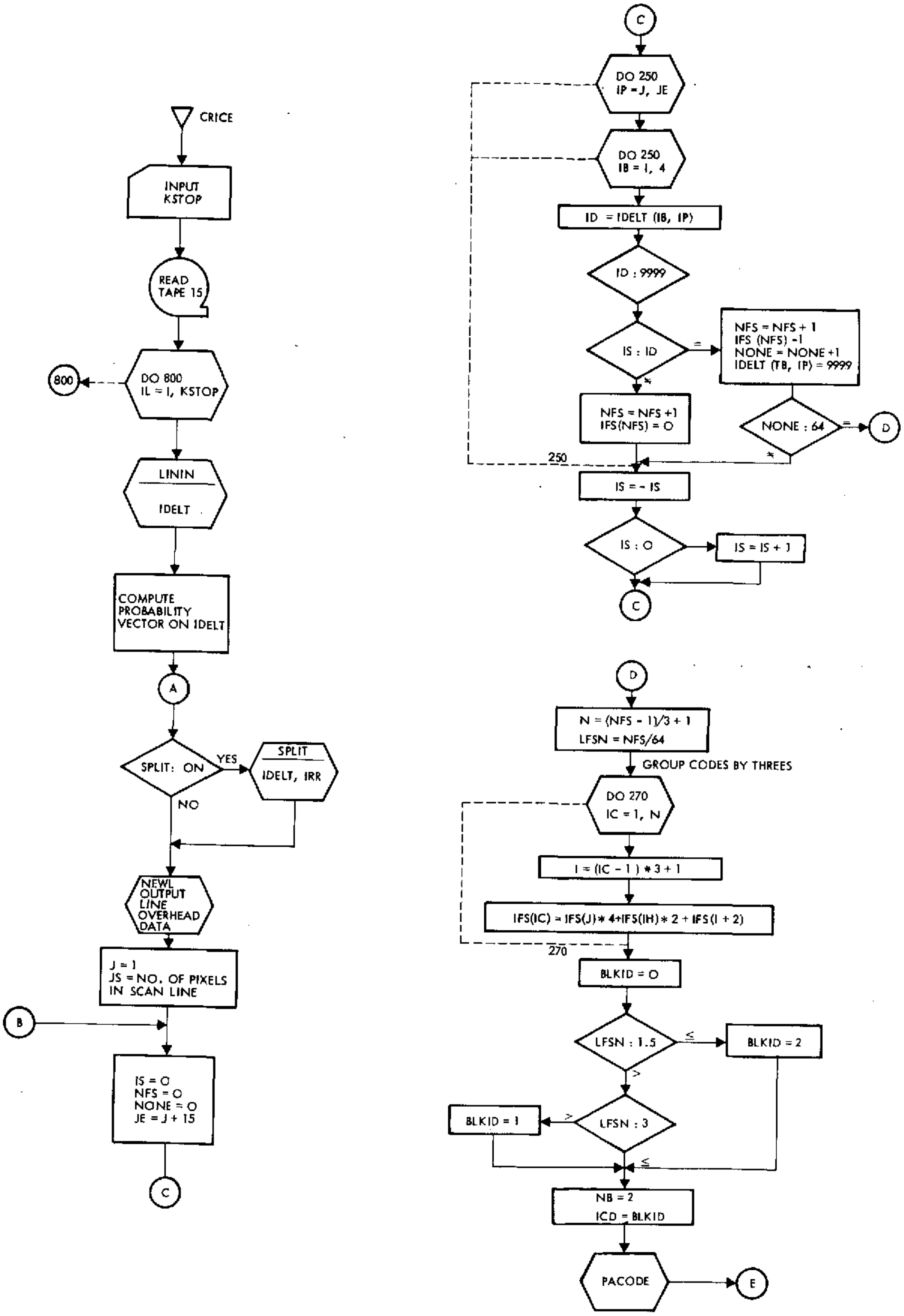


Figure 2.4-14. Flow of Program CRICE

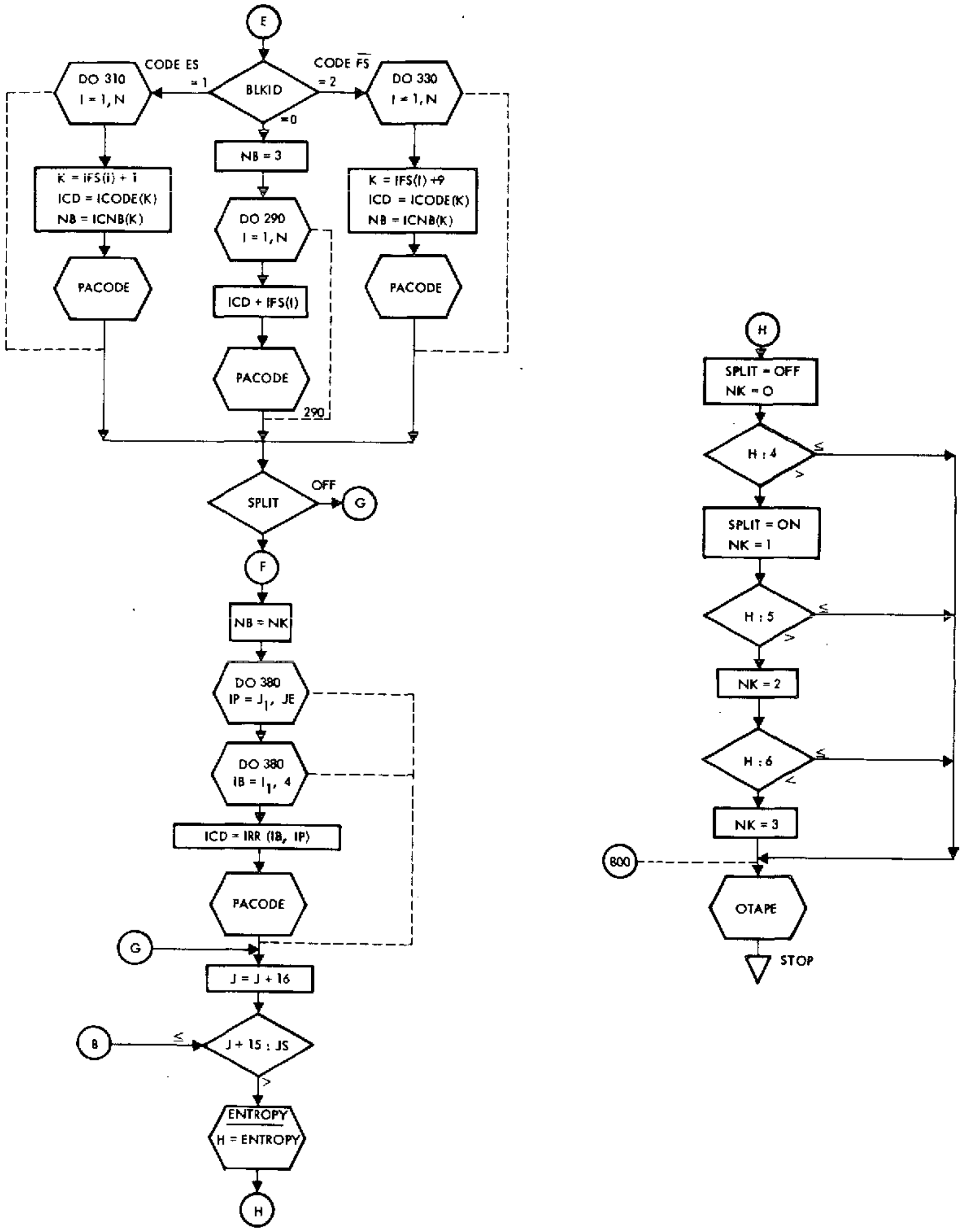


Figure 2.4-14. Flow of Program CRICE (Continued)

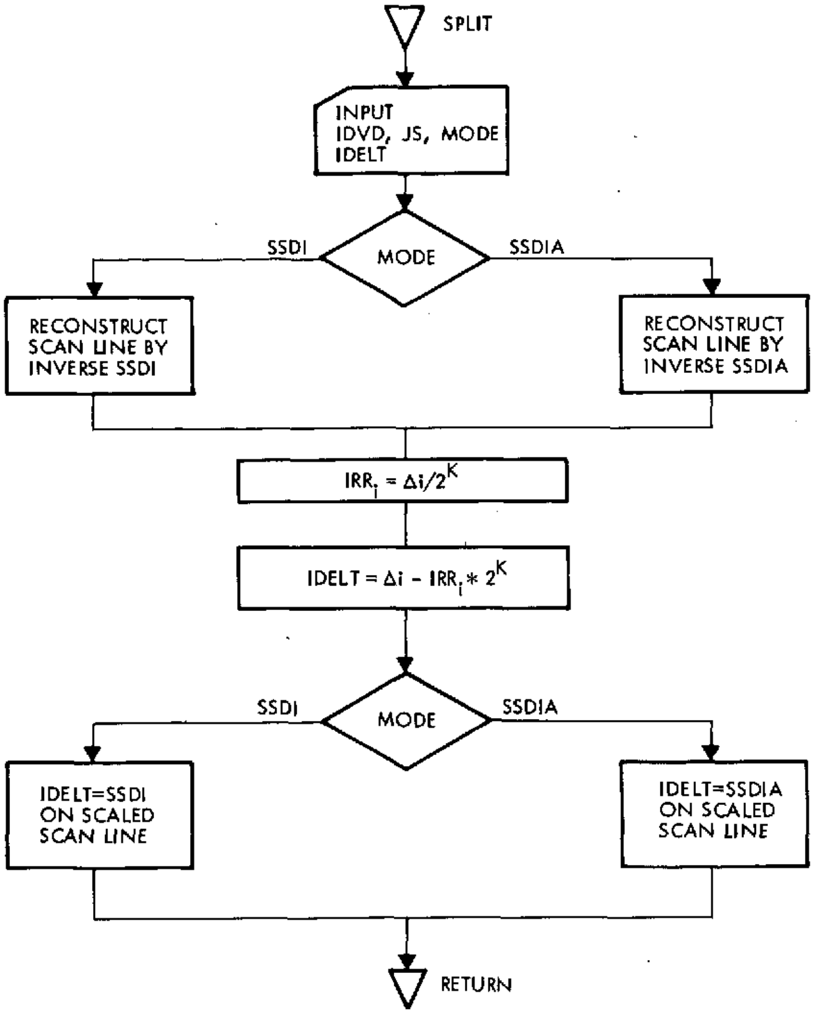

SPLIT
INPUT
IDVD, JS, MODE
IDELT
MODE
SSDI
SSDIA
RECONSTRUCT SCAN LINE BY INVERSE SSDI
RECONSTRUCT SCAN LINE BY INVERSE SSDIA
$IRR_i = \Delta i/2^K$
$IDELT = \Delta i - IRR_i * 2^K$
MODE
SSDI
SSDIA
IDELT=SSDI ON SCALED SCAN LINE
IDELT=SSDIA ON SCALED SCAN LINE
RETURN

Figure 2.4-15. Flow of Subroutine SPLIT

Subroutine NEWL outputs the two bits of overhead data required at the start of each new scan line to designate split-pixel mode used and the initial pixel intensity values. Loop 250 computes the fundamental sequence for each block of 16 pixels (64 symbols) in the scan line by simulating the wiggle operation given in 2.3.1. The number of bits in the sequence NFS is normalized by the number of samples to obtain LFSN. The fundamental sequence, augmented if necessary, is sub-divided into groups of three bits.

Based on the value of LFSN, the fundamental sequence is transmitted directly (BLKID = 0), coded directly (BLKID = 1), or complemented and coded (BLK(D = 2). At the beginning of each data block PACODE generates the block overhead bits required to specify the Rice mode used. After the fundamental sequence has been encoded as required these bits are then packed by PACODE. If the split-pixel mode is operable for that block, flow proceeds to F where the residuals IRR are sequentially packed by PACODE.

The above operations are performed block-by-block until the end of the scan line is encountered. At this point the entropy H is computed for that scan line to determine which split-pixel mode, if any, is optimal for use on the following scan line (see 2.3.2). If not the last scan line, the program flow returns through loop 800 to process the following scan line. After the last scan line has been processed, subroutine OTAPE empties the output tape buffer and program CRICE terminates.

2.4.7 Program RSTRIC

RSTRIC reconstructs the Rice encoded data packed on RICEP by program CRICE. The flow of RSTRIC is given in Figure 2.4-16. The operation of this program is of greater complexity than that of RSTHUF used for reconstructing Huffman encoded data since in addition to determining the compression mode of the data (SSDI or SSDIA) the reconstruction mode must be determined for each block of data. As illustrated for a block of 16 pixels (64 intensity samples), each block can be encoded by transmitting the fundamental sequence (FS), the coded fundamental sequence (CFS), or by complementing and coding the fundamental sequence ($\overline{\text{CFS}}$). In addition, each line can be transmitted in the split-pixel mode. These various modes are communicated to the reconstruction algorithm in the form of line and block overhead bits given by the format included in section 2.4.6.

Input parameter NLINE specifies the number of lines to be reconstructed within loop 800. Subroutine INTAP reads the first record from the input tape of encoded data and GET5, similar in function to GET12 of program RSTHUF, obtains the first five bits in IWD to determine the new line identification and the mode (SSDI or SSDIA) to be used for reconstructing the data. The following 28 bits are used to establish the initial intensities in each spectral band. Following this, the next two bits give the split-pixel mode used for compressing that line of data. After this initialization of line operations, block reconstruction is performed until the next sequence of bits occur denoting the start of a new scan line.

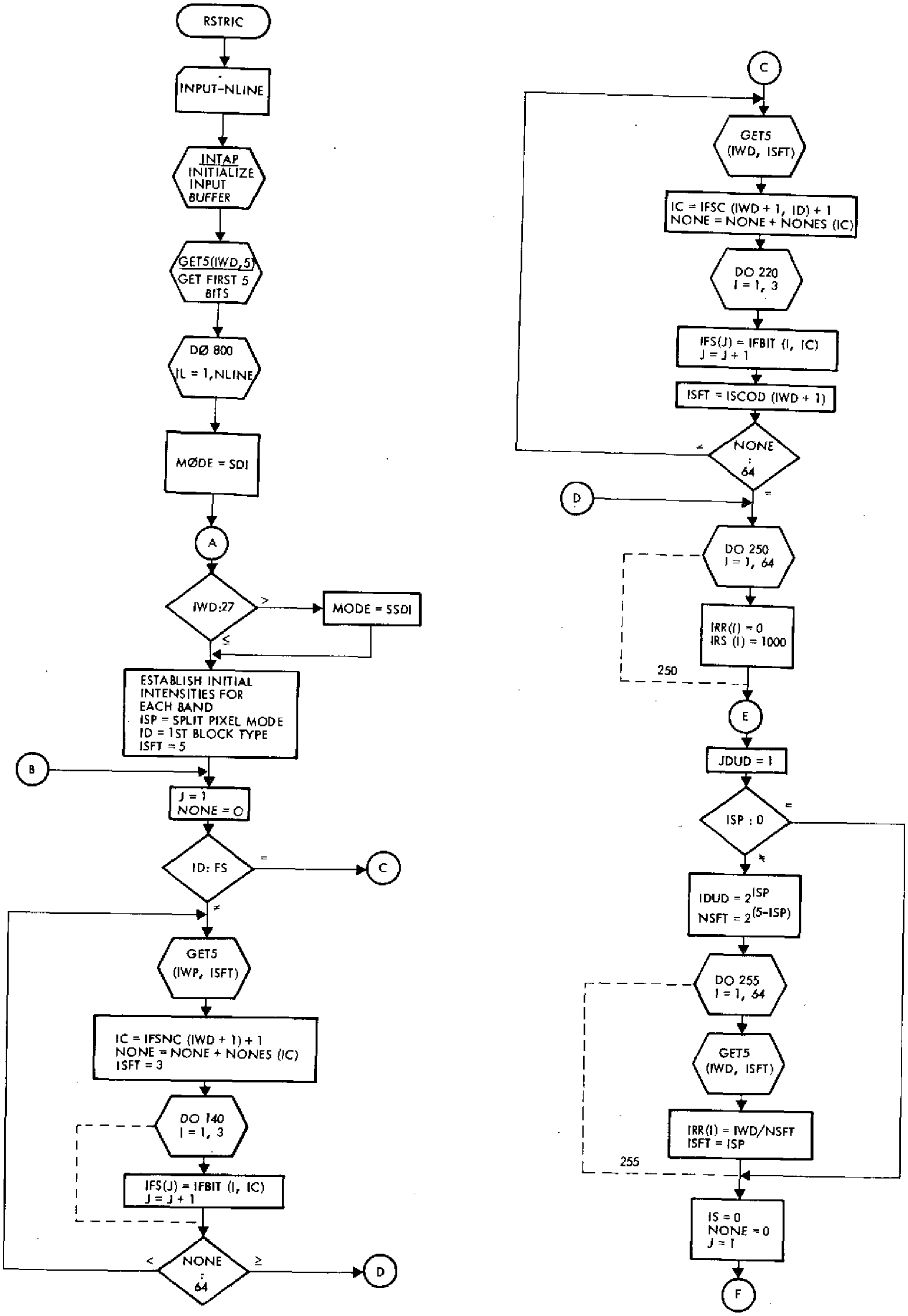


Figure 2.4-16. Flow Diagram of Program RSTRIC

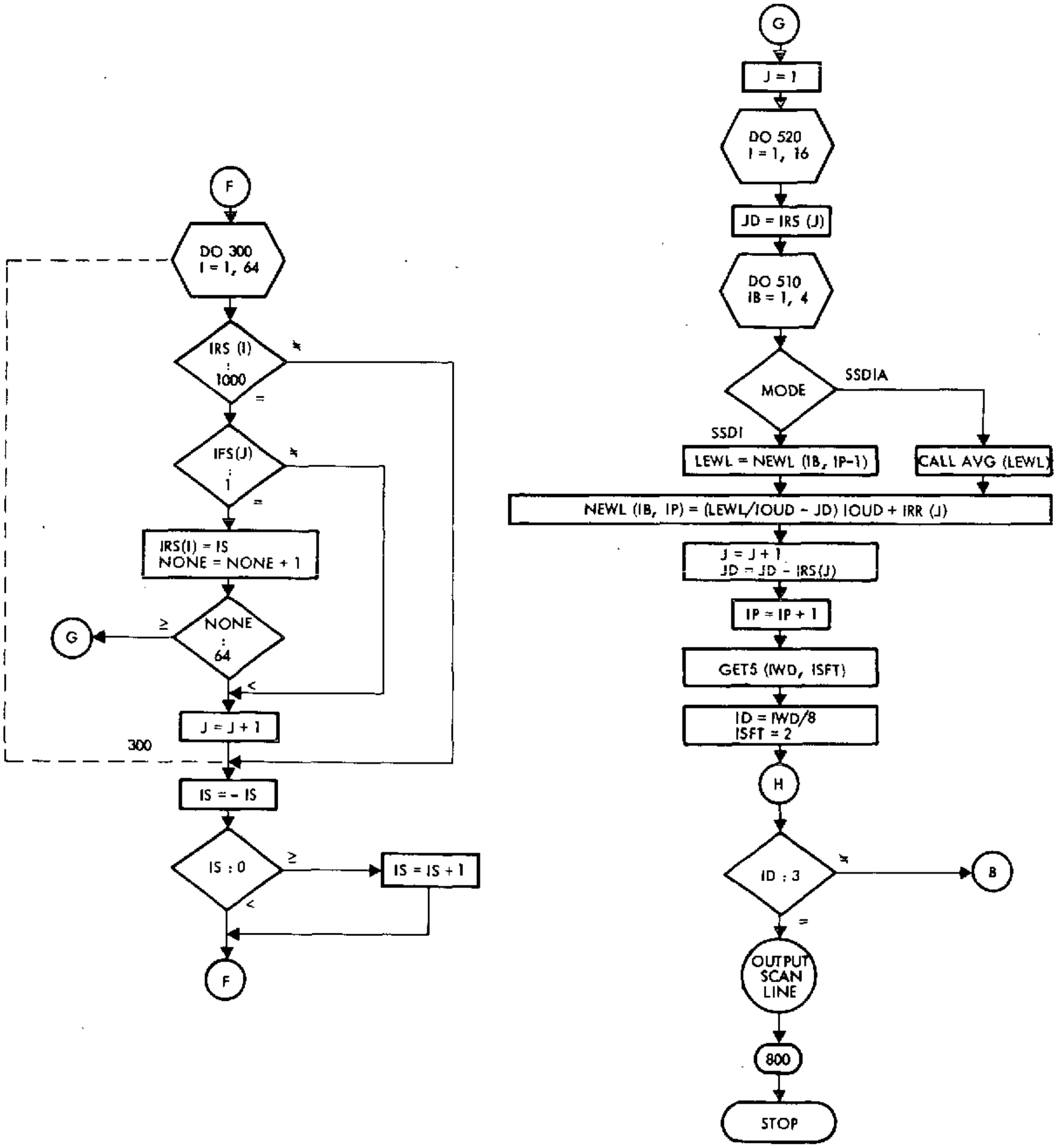


Figure 2.4-16. Flow Diagram of Program RSTRIC (Continued)

The first two bits of block information denote which of the three Rice modes are used for that block. These bits are denoted by ID; if ID = 0, the FS is used, if ID = 1, the coded FS is used, and if ID = 2, the complemented and coded FS is used. If ID = 1 or ID = 2, flow goes to point C to regenerate the fundamental sequence. Five compressed bits are fetched at a time in IWD to address array IFSC which contains the decoding table. IC is the index to the code word sent and NONE counts the number of one bits decoded in the fundamental sequence.

Vector IFS contains the regenerated fundamental sequence, complemented if ID = 2, and ISFT contains the number of bits in the current codeword. To decode the next codeword the compressed bit sequence is shifted by ISFT bits and the following five bits are extracted. This procedure continues until there are 64 one bits in the decoded fundamental sequence, implying that all codewords in the block have been decoded. If ID = 0, the fundamental sequence is regenerated directly in IFS by extracting the compressed bit string until IFS contains 64 one bits.

Following reconstruction of the fundamental sequence in IFS, flow proceeds to D and loop 250 which initializes the 64 compressed symbols (IRS) to be reconstructed and the 64 residuals (IRR) used for the split-pixel mode. If a split-pixel mode (n, k) is used, loop 255 extracts the following 64k bits to regenerate these residuals in vector IRR.

Loop 300 regenerates the vector IFS of symbols generated by the SSDI or SSDIA algorithms. The technique used to generate the fundamental sequence involved wiggling through the symbol levels in the order, 0, 1, -1, 2, -2, 3, etc. and the inverse of this operation is performed to obtain these symbols in IRS. Parameter IS, initialized to zero, keeps track of the symbol value being reconstructed in each pass. Initially, the first 64 bits of the fundamental sequence are tested and the presence of a one bit at location I sets entry IRS(I) = 0. This first pass reconstructs all symbols having a zero value. On successive passes, the bits IFS are tested and the presence of a one bit in location I sets IRS(I) = IS, the symbol level currently being generated. Since the entries in vector IRS were initialized to 1000, the presence of any other value in an entry of IRS signals that this symbol has already been reconstructed and this symbol is skipped on succeeding passes. Each occurrence of a one bit in the fundamental sequence increments parameter NONE. When NONE = 64, all symbols have been reconstructed for that block and flow exits loop 300 to point G which contains the inverse SSDI or SSDIA algorithm, depending on the mode used for compressing the data. Loop 510 reconstructs each set of four intensities based on the symbols IRS and residuals IRR which have been regenerated from the compressed data. Loop 520 reconstructs the sixteen pixels contained in the current block. If the SSDIA mode is used, LEWL contains the preceding reconstructed pixel intensities in that scan line. If the SSDIA mode is used, LEWL contains the average of the four appropriate reconstructed intensities, as obtained by subroutine AVG.

After reconstruction of a block, GET5 extracts the next five bits and tests the following two bits to determine whether the following compressed data represents another block in the same scan line or the start of a new scan line. If a new scan line, the current reconstructed scan line of data is output on a tape and flow proceeds within loop 800 until all NLINE scan lines have been reconstructed.

Provision is made through DATA statements for varying block sizes, split-pixel modes, and selection of code words for the fundamental sequence in the event that the operator decides to change these parameters of the Rice algorithm.

LID	UNCODED FIRST ELEMENT	BID	RICE CODED n BITS	k BITS
2 Bits	28 Bits	2 Bits	Variable	Lk bits

REFERENCES

1. Huffman, D.A., "A Method for the Construction of Minimum Redundancy Codes," Proc. IRE, Vol. 40, No. 10, pp. 1098-1101, Sept. 1952

2. Rice, R.F., "The Rice Machine: Television Data Compression," GTD 900-408, JPL, Pasadena, California, Sept. 1, 1970

3. RESULTS OF THE INVESTIGATION

This section discusses the significance of the various measurements performed on the ERTS tapes, the selection of scenes processed, and summarizes the results obtained for both normal ERTS data and for imagery containing anomalous data. Conclusions and tradeoffs based on these results are discussed further in section 4.

3.1 SUMMARY OF PROCESSING PERFORMED

During the data analysis phase of the study, thirty subscenes and four full scenes were processed and the following output quantities were obtained for each scene:

1. MSS Data Statistics
   a. Data mean and variance per spectral bands and average overall bands.
   b. Cross spectral-spatial correlation
   c. Spectral correlation (joint probability density function)
2. Data Compression Performance
   a. Probability distribution function of first differences obtained by the SSDI, SSDIA, and SSDIAM modes.
   b. Probability distribution function of the Shell, SSDI, SSDIA, and SSDIAM symbols.
   c. Compression achieved by fixed Huffman coding of the scene using the Shell, SSDI, SSDIA, and SSDIAM modes.
   d. First-order entropy of these distributions.
   e. Line-by-line and overall time-varying data compression using the fixed Huffman, adaptive Huffman, and Rice algorithms on the selected compression mode.
   f. Buffering statistics of the Huffman or Rice Code.
   g. Huffman codes associated with each of the compression modes.

All four full scenes were compressed and reconstructed using the SSDIA/Huffman or SSDI/Rice algorithm and photographs were made from the reconstructed data. In addition to the above outputs, the following data have been obtained for the principal full scene, ERTS 1015-17440:

a. Photographs of the reconstructed image obtained by use of the essentially information preserving SSDIAM algorithm with mappings of ± 1 level and ± 3 levels.

b. Photographs of reconstructed images with the compressed data corrupted by simulated channel errors (10^{-5} and 10^{-6} bit error probabilities).

c. Probability distribution of the original intensity levels of the MSS data.

3.2 SCENES AND OBJECT CLASSES PROCESSED

During the investigation thirty 5 x 5 nautical mile square subscenes and four 25 x 25 nautical mile square full scenes were processed. The selection of scenes to be processed were based on several criteria including availability from NASA-ERTS User Services. A minimal set of nine object classes was specified in the data analysis plan. This set of object classes is as follows:

1. Clouds
2. Bodies of Water (Lakes, Oceans)
3. Rivers
4. Snow
5. Mountains
6. Agriculture
7. Plains
8. Deserts
9. Forests.

These classes were based on coverage of the predominant objects encountered on earth survey missions rather than selection by the criteria of usefulness to current principal investigators. Classes were further chosen to span the range of source data activity, thereby serving to roughly bound the expected compressed data rates which could occur.

In addition to the nine classes given above, the object classes of "cities" and "grassland" were added. In addition to the homogeneous object classes, several additional composite classes were selected corresponding to class variants which are also commonly encountered and which would be expected to yield different data and compression statistics. These object classes and the basis for their selection are detailed below:

- Coastline and Harbor - These features are encountered at the boundary between land and bodies of water. Since the data statistics and characteristics of these two features vary greatly, it is of interest to determine what effect the land-water interface has on the various compression algorithms, especially for the global Huffman codes which develop a code optimal for neither the water nor the land.
- Island - Similar to the coastline class, this object is of primary interest in the determination of the time-varying bit rate and buffer statistics as the data activity varies from very low (ocean) to high (island).
- Haze - The effect of haze obscuring the land features amounts to a decrease in contrast and data activity compared to the scene without haze. The predominant characteristic sought is the decrease in bit rate produced by such haze.
- Scattered Clouds over Water - This class produces a wide distribution of difference intensities in the compressed symbols due to the large intensity steps between the bright clouds and dark water. Processing this class serves to evaluate the ability of the compression algorithms to hardle such symbol distributions.

Certain classes subject to wide variations were processed as conditions charged. As an example, three variants of the object class containing mountains were selected; bare mountains, mountains with vegetation, and mountains with snow cover. As expected, the results differ substantially. Several scenes containing agriculture were processed, corresponding to differing crops and field sizes.

The statistics and compressed rate can vary on any given object class and for any given location depending on varying factors such as the time of year, sun angle, and cloud cover. While it was not possible to obtain results for all such variations within the scope of the current investigation, it is felt that the results which have been obtained are representative of each class and serve the objectives of the study.

The object class subscenes were chosen to encompass an area of 25 square miles since the chosen object classes can be located to span such an area and such a size yields statistically significant results. The 625 square mile full scenes were selected either to contain a suitable grouping of object classes or to ascertain whether the data and compression statistics of an object class covering such an area would deviate significantly from the statistics of a subscene containing that class. Emphasis was given to subscene processing since the degree of compression expected for a given full scene can be estimated by knowing the percentage occurrence of the various object classes in the full scene and the average compressed bit rate obtainable for the various object classes, as determined by subscene processing. Conversely, since full scenes normally contain several object classes, it is not possible to extrapolate the bit rate and statistics measured for the full scene to subscenes and object classes contained within it.

After selection of object classes, tapes were selected from supplied imagery to include the various required subscenes and full scenes processed during the study. Each entry of Table 3.1 gives the scene identification number, object class description, location, earth coordinates, ERTS tape number, and the location (in pixels) of the upper lefthand corner of the scene processed. Note that scenes one through thirty are subscenes and numbers thirty-one through thirty-four are full scenes. These scene identification numbers will be used throughout this report.

Table 3.1. Parameters of Scenes Processed

SCENE NO.	CLASS DESCRIPTION	LOCATION	COORDINATES(N,W)	TAPE NO.	STRIP	(STARTING PIXELS)
1	CLOUDS	KANSAS	(36',40"; 101',10")	1043-16573	4	(230,180)
2	BAY	CHESAPEAKE BAY	(38',05"; 76',15")	1062-15190	4	(1900,420)
3	LAKE	LAKE MICHIGAN	(43',15"; 87',15")	1017-16093	4	(220,370)
4	OCEAN	PACIFIC OCEAN	(34',15"; 119',30")	1018-18010	1	(1300,10)
5	LAKE	LAKE ST. JOHN, CAN.	(48',35"; 72',00")	1025-15103	2	(880,420)
6	SNOW	VERMONT	(46',10"; 73',10")	1170-15173	4	(400,420)
7	COASTLINE	MASS., N.H.	(42',45"; 70',45")	1167-15011	3	(1080,350)
8	CLOUDS (OVER FOREST)	QUEBEC, CANADA	(49',20"; 70',30")	1025-15103	4	(1800,301)
9	DESERT	IMPERIAL VALLEY, CA.	(32',30"; 113',50")	1015-17440	4	(2050,600)
10	HARBOR	LONG BEACH, CA.	(34',05"; 117',30")	1018-18010	4	(1850,20)
11	PLAINS	TEXAS PANHANDLE	(35',40"; 103',00")	1043-16573	1	(1700,540)
12	CLOUDS (OVER OCEAN)	SOUTHERN CALIFORNIA	(33',45"; 119',30")	1018-18010	1	(2010,420)
13	FOOTHILLS	SAN BERNARDINO, CA.	(33',15"; 113',45")	1106-17501	4	(1000,301)
14	FOREST	VIRGINIA	(38',00"; 77',30")	1062-15193	2	(300,270)
15	ISLAND	CATALINA, CA.	(33',25"; 118',30")	1108-18020	3	(510,600)
16	HAZE (OVER MOUNTAINS)	QUEBEC	(45',50"; 75',20")	1170-15173	2	(1000,440)
17	CITY	LOS ANGELES, CA.	(34',00"; 118',10")	1018-18010	3	(1700,600)
18	DESERT	MOJAVE DESERT	(34',55"; 117',55")	1018-18010	3	(150,600)
19	(BARE) MOUNTAINS	SAN BERNARDINO, CA.	(33',15"; 114',25")	1015-17440	3	(1300,10)
20	CITY	CHICAGO	(41',50"; 87',40")	1017-16093	4	(1800,500)
21	GRASSLAND	NEBRASKA	(41', 0"; 101',05")	1007-16560	2	(5,50)
22	FOREST	QUEBEC, CAN.	(49', 0"; 72',20")	1025-15103	1	(500,610)
23	(RIVERS (IN FOREST)	QUEBEC, CAN.	(47',50"; 73',05")	1025-15103	1	(2095,150)
24	AGRICULTURE	NORTH KANSAS	(39',50"; 101',25")	1007-16560	2	(1800,500)
25	AGRICULTURE	TEXAS/OKLA.	(36',20"; 103',05")	1043-16573	1	(1000,100)
26	(VEGETATED) MOUNTAINS	SOUTHERN CALIFORNIA.	(34',15"; 117',45")	1018-18010	4	(1070,370)
27	AGRICULTURE	ILLINOIS	(41',50"; 88',40")	1017-16093	2	(1990,1)
28	(SNOW IN) MOUNTAINS	LAKE TAHOE, CA.	(38',35"; 119',55")	1128-18120	3	(1420,330)
29	FOREST	WISCONSIN	(43',15"; 88',36")	1017-16093	2	(50,50)
30	AGRICULTURE	IMPERIAL VALLEY, CA.	(32',45"; 115',30")	1015-17440	1	(2000,100)
31	PRINCIPLE FULL SCENE	IMPERIAL VALLEY, CA.	(33',40"; 114',35")	1015-17440	2	(880,1)
32	FOREST	QUEBEC, CANADA	(49',00"; 72',00")	1025-15103	2	(290,1)
33	MOUNTAINS	SOUTHERN, CALIFORNIA	(34',35"; 119',00")	1018-18010	1	(800,1)
34	DESERT	MOJAVE DESERT	(34',50"; 118',05")	1018-18010	4	(4,1)

3.3 ERTS IMAGERY CHARACTERISTICS AND STATISTICS

3.3.1 Significance of Statistical Measurements Used

The performance of data compression algorithms such as the SSDI and SHELL techniques used in this study is highly dependent on the characteristics of the source data. The compression achieved on a scene is proportional to the degree of spectral and spatial correlation existing in the data. For this reason, knowledge of the statistics of the multispectral scanner data is important.

Several data measurements were performed for all scenes evaluated. Initially the mean and variance of the data are obtained for each spectral band and subsequently averaged over all bands. The mean intensity level of each spectral band level is averaged over the scene to indicate the average distribution of spectral energies in the scene. The variance of each band corresponds to the degree of data activity within that spectral band as averaged over the scene.

Neither the mean nor the variance are accurate indicators of the compression that can be achieved for the scene, although a very low variance indicates high compression and a very high variance normally corresponds to a low degree of compression. Since the compression algorithms are essentially forms of differential pulse code modulation (DPCM), the effects of the differing spectral means are eliminated. Since the algorithms are based only on localized data activity it is possible for the intensity levels in each band to vary greatly over different portions of the global scene, yielding a large data variance, while varying slowly over local areas of the scene. Thus, overall variance of the intensity within the spectral bands is only a weak indicator of compression performance.

The cross-spectral-spatial correlation is a more accurate indicator of the compression performance for a scene because it measures the localized changes in the spectral information. Each ground picture element corresponds to a four-dimensional intensity vector $\underline{I}$ constructed from the intensity values of the four spectral components corresponding to that element. This vector moves through the spectral subspace having a unique location for each ground element. If the data activity is low in a given region, the vector movement is small between adjacent pixels and, conversely, the vector can undergo large excursions between pixels in regions of high data activity.

For the algorithms used in this study, compression is not only dependent on the magnitude and variability of the vectors from pixel to pixel but on the type of the movement. If the vector direction remains unchanged from pixel to pixel and only the magnitude varies, the compression can remain high since this corresponds to a high spectral correlation between pixels. Conversely, changes in vector direction with unchanged magnitude corresponds to low spectral correlation and henceforth less compression.

The cross spectral-spatial correlation was developed to correspond to this dependence of the algorithm on the data activity. For each pair of intensity vectors $\underline{I}_i$ and $\underline{I}_{i+k}$ in the scene, where k is spatial separation of the elements, the normalized dot product is formed. Normalization removes the effect of scene illumination and the effects of magnitude changes while the dot product reflects the variation in vector direction between these elements. The closer this normalized dot product to unity, the greater the correlation between pairs and the better the expected performance of the algorithms. The curves generated for each scene reflect the average correlations obtained. The abscissas of the curves give the element spacing in pixels, the ordinates represent normalized cross spectral-spatial correlation, σ_k, and the curves give the percent of vector pairs in the scene which have a correlation greater than σ_k for the spacing k.

The joint probability distribution function measures the joint occurrence of first difference values over the scene. A four-dimensional difference vector $\underline{\Delta}$ is formed by subtracting each pixel intensity from the intensity of the preceding pixel along the scan line. Vector $\underline{\Delta}$ corresponds to the first set of differences as obtained by the SSDI algorithm. Each difference vector corresponds to a point in a four-dimensional space. By counting the occurrence of these vectors over the scene and converting to a percentage occurrence, the clustering of the differentials can be observed. The location and degree of clustering of these difference vectors about the origin (0,0,0,0) gives another indication of achievable compression. Because a four-dimensional plot cannot be performed, only the two-dimensional projections of the probability space is given and all $\binom{4}{2}$ = 6 projections are printed by the computer. In order to produce an intelligible output, the occurrence of all zero difference values, the most probable occurrence, is normalized to the value 100 and all other probabilities are normalized to this value. Therefore, if the occurrence of al +1 difference values is assigned the value 60, this means that P(1,1)/P(0,0) = .6. If a joint occurrence of a set of differences occurs less than 1% as often as does the joint occurrence of zeros, no value is printed for that location in order to produce a clean display.

3.3.2 Results Obtained

This study produced well over 1000 pages of computer printout. A complete set of output is given in Appendix B for the principal full scene. This section summarizes the various statistical measurements obtained for the thirty-four scenes by presenting results typical of the range of data generated, including anomalous data.

Figure 3.3-1 gives the probability density of the data intensities present in each band of the principal full scene (number 31). The means and variances of this data are:

	Band 1	Band 2	Band 3	Band 4	Average
Mean	51.220	57.761	56.890	23.880	47.438
Variance	174.436	273.124	170.609	43.519	356.721

Note that the average variance is not the average of the variances in each spectral band. This measure represents the overall variance of the data in all spectral bands based on the average mean of the four bands. It is normal for this variance to be greater than the variances of the individual bands, as in the case above.

Note that the probability density function of the MSS data is not continuous as might be expected in spectral bands 1, 2, and 3, and contains intensity levels having much lower occurrence than adjacent levels. This anomaly arises from the decompression algorithms used for ground processing of the received data. The effect of this intensity mapping is quite evident in the probabilities of levels 56 through 64 of band 2. The distribution of intensities in band 4 (infrared) does not exhibit this peculiarity since no mapping is performed for that band. This mapping has an adverse effect on

LEVEL	BAND1	BAND2	BAND3	BAND4
0	0.0000	0.0000	0.0000	0.0000
1	C.0000	0.0000	0.0000	0.0000
2	C.0000	0.0000	0.0000	.0195
3	.0002	0.0000	0.0000	.0800
4	0.0000	0.0000	0.0000	.1027
5	C.0000	0.0000	0.0000	.0527
6	0.0000	0.0000	0.0000	.0637
7	C.0000	0.0000	0.0000	.0997
8	0.0000	0.0000	0.0000	.1777
9	C.0000	0.0000	0.0000	.2677
10	0.0000	0.0000	.0027	.4310
11	0.0000	0.0000	.0120	.5760
12	0.0000	0.0000	.0367	.7955
13	0.0000	.0012	.0602	1.0525
14	C.0000	.0150	.0395	1.4400
15	C.0000	.0205	.0257	1.6867
16	C.0000	.0185	.0435	2.9120
17	C.0000	.0357	.0352	3.7430
18	C.0000	.0720	.0420	5.0185
19	0.0000	.0977	.0365	6.1420
20	0.0000	.1367	.0250	6.4765
21	.0002	.3152	.0712	6.2447
22	.0075	.1425	.0927	5.7335
23	.0250	.9715	.0555	4.7267
24	.0537	1.1967	.1542	5.9690
25	.0762	.6345	.1802	4.9340
26	.0815	.6165	.0790	4.9672
27	.0600	.5717	.2727	4.8477
28	.2640	.6102	.1962	4.6322
29	.2972	.9897	.4035	4.8322
30	.2102	.7240	.4060	4.9772
31	.6637	.8005	.2032	4.1320
32	1.6167	.4475	.4755	3.6617
33	.9960	.9032	.2535	2.0167
34	1.8085	.5222	.6092	1.2280
35	1.9405	.5192	.3085	.8602
36	1.5612	1.0210	.8130	.6937
37	2.8132	.4417	.3152	.6330
38	1.5137	1.3000	1.1625	.5562
39	2.1930	.2922	.5320	.5105
40	2.5570	1.5357	1.2522	.4767
41	1.9010	1.5790	1.0105	.4137
42	3.1440	.9380	2.0887	.3650
43	2.1617	1.3037	2.1877	.3307
44	4.2600	1.0677	1.6450	.3027
45	2.2107	1.4832	2.2867	.2460
46	3.8215	1.3255	2.9222	.2080
47	4.2705	1.7122	1.9735	.1472
48	2.6060	1.9722	3.5762	.0882
49	4.7780	1.3870	2.1565	.0512
50	1.2317	3.1187	2.9935	.0295
51	4.3165	1.7247	3.4140	.0202
52	3.2300	2.9577	2.5920	.0132
53	3.9825	2.8107	4.1755	.0085
54	2.3770	3.1760	.0537	.0030
55	3.1960	3.1520	4.6037	.0010
56	1.4120	3.8102	2.3690	0.0000
57	2.0375	6.4202	3.8682	0.0000
58	2.3045	.7627	3.0022	0.0000
59	1.5855	1.4105	.8360	0.0000
60	3.2720	.7652	3.4582	0.0000
61	1.5070	3.2942	2.1807	0.0000
62	2.5765	.7947	4.1015	0.0000
63	1.0690	3.1830	2.0962	0.0000
64	1.7467	1.2592	0.0000	0.0000
65	3.3257	1.6485	3.9587	0.0000
66	.8387	1.5455	1.9707	0.0000
67	1.9222	2.1825	.0581	0.0000
68	1.9792	.7192	3.5847	0.0000
69	1.6735	1.4315	1.7360	0.0000
70	1.8690	1.9160	.9345	0.0000
71	1.9657	.8825	2.1955	0.0000
72	.6480	1.3352	1.4980	0.0000
73	.5537	1.1915	1.2115	0.0000
74	.6032	1.7340	1.7585	0.0000
75	.0867	1.8720	2.3132	0.0000
76	.1970	.4825	1.3025	0.0000
77	.0957	1.2097	1.7967	0.0000
78	.0465	1.5847	2.2352	0.0000
79	.0475	1.0465	.6537	0.0000
80	.0510	.5797	1.0955	0.0000
81	.0355	1.5997	1.8957	0.0000
82	.0265	.5807	.2700	0.0000
83	.0460	1.1727	.6600	0.0000
84	.0187	1.6642	.3750	0.0000
85	.0247	.4107	.1182	0.0000
86	.0062	1.3052	.1462	0.0000
87	.0457	1.2400	.1112	0.0000
88	.0022	.4265	.0435	0.0000
89	.0192	.8260	.0532	0.0000
90	.0235	1.2587	.0440	0.0000
91	.0097	.2307	.0272	0.0000
92	.0167	.5285	.0352	0.0000
93	.0060	.5402	.0242	0.0000
94	.0165	.1142	.0085	0.0000
95	.0012	.1112	.0270	0.0000
96	.0122	.0802	.0112	0.0000
97	.0082	.0275	.0047	0.0000
98	.0072	.0355	.0120	0.0000
99	.0047	.0240	.0075	0.0000
100	.0060	.0042	.0035	0.0000
101	.0020	.0155	.0065	0.0000
102	.0022	.0215	.0035	0.0000
103	.0060	0.0000	.0040	0.0000
104	.0015	.0045	.0047	0.0000
105	.0010	.0212	.0025	0.0000
106	.0015	0.0000	.0017	0.0000
107	.0017	.0030	.0002	0.0000
108	0.0000	.0027	.0015	0.0000
109	.0012	.0067	.0005	0.0000
110	.0005	.0037	.0007	0.0000
111	.0005	.0030	.0007	0.0000
112	.0005	.0025	0.0000	0.0000
113	.0002	.0015	.0010	0.0000
114	0.0000	.0010	0.0000	0.0000
115	.0012	.0020	0.0000	0.0000
116	0.0000	0.0000	.0002	0.0000
117	C.0000	.0020	0.0000	0.0000
118	0.0000	.0002	.0005	0.0000
119	.0002	0.0000	.0002	0.0000
120	0.0000	.0002	0.0000	0.0000
121	0.0000	.0010	.0002	0.0000
122	0.0000	0.0000	.0007	0.0000
123	.0002	.0005	0.0000	0.0000
124	C.0000	.0002	0.0000	0.0000
125	0.0000	0.0000	0.0000	0.0000
126	.0005	.0005	0.0000	0.0000
127	0.0000	.0005	0.0000	0.0000

Figure 3.3-1. Probability of Occurrence of Intensity Levels in Each Band of Principal Full Scene (31)

the SSDI algorithms which are based on differences between successive intensity levels and the mapping algorithm introduces another source of noise into the data. The SSDIA and the SSDIAM algorithms tend to smooth these fluctuations due to the averaging operations in the SSDIA and the remapping ability of the SSDIAM.

Figures 3.3-2 and 3.3-3 show the probability density of the data intensities for scene 28. Figure 3.3-2 is based on data received from the spacecraft while Figure 3.3-3 is based on the data after ground processing. The effect of the ground mapping algorithms is evident in a comparison of these two figures.

A second anomaly present in several of the subscenes processed is due to the presence of a bad sensor in band 2 (MSS-5). This defect produces abnormally low intensity values on every sixth scan line of data in band 2. Figure 3.3-4 gives a segment of input data which clearly illustrates the occurrence beginning on the fourth scan line shown and recurring every sixth line. The effect of this anomalous data on the compression algorithms will be discussed further in Section 3.4.2.

Figures 3.3-5 through 3.3-8 show several plots of cross spectral-spatial correlation for processed scenes. Figures 3.3-5 is from scene number 4 (ocean) and shows highly correlated data since the spectral vector undergoes only minor changes in direction over the scene. As expected from such a high degree of correlation, a very low compressed bit rate was produced for this scene. Figure 3.3-6, from scene number 22 (forest) illustrates a low degree of correlation implying a large average change in vector direction for even closely spaced pixels. Figures 3.3-7 and 3.3-8 give the 95% correlation curves for several additional object classes.

Similarly, Figures 3.3-9 through 3.3-11 show several joint probability distributions of first difference as obtained by the SSDI algorithm to illustrate several types of clustering which occurred. Figure 3.3-9 illustrates a high degree of clustering of joint differences about (0,0) in spectral bands 1 and 2, based on scene 4 (ocean). Figure 3.3-10 illustrates a wide spread of differences as produced by scene 23 having higher data activity. Figure 3.3-11 shows an intermediate case from scene 29.

LEVEL	BAND1	BAND2	BAND3	BAND4
0	0.0000	0.0000	0.0000	0.0000
1	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000	0.0000
4	0.0000	0.0000	0.0000	.0482
5	0.0000	0.0000	0.0000	.0413
6	0.0000	0.0000	0.0000	.0172
7	0.0000	0.0000	0.0000	.0103
8	0.0000	0.0000	0.0000	.0138
9	0.0000	0.0000	0.0000	.0138
10	0.0000	0.0000	0.0000	0.0000
11	0.0000	0.0000	0.0000	.0310
12	0.0000	0.0000	.0103	.0722
13	0.0000	0.0000	.0482	.2202
14	0.0000	.0069	.0206	.9460
15	0.0000	0.0000	.0069	1.9917
16	0.0000	.0275	.0069	6.3330
17	0.0000	.0344	.0103	9.6732
18	0.0000	.2339	.0069	11.7165
19	0.0000	1.0870	.0034	13.0478
20	0.0000	2.1156	.0103	12.8552
21	0.0000	4.0213	.0103	12.1465
22	.0172	3.3196	.0034	9.9140
23	.0791	5.7241	.0103	6.4706
24	.3337	8.1493	.0172	5.4008
25	2.9309	10.0378	.0310	3.0100
26	2.8070	5.8789	.0550	1.9229
27	6.0028	.9563	.1342	1.1799
28	8.0323	8.9783	.5607	.7671
29	7.8569	3.2439	.8256	.6674
30	15.4283	7.3203	2.3667	.4334
31	6.4809	4.8435	4.6096	.2614
32	7.9842	1.5136	1.5445	.2718
33	5.8927	2.6660	7.6127	.2098
34	6.8352	2.3117	2.8208	.1376
35	1.3691	2.2876	9.3705	.0791
36	5.4352	2.1603	3.3987	.0516
37	4.5889	1.9780	9.3774	.0172
38	2.2910	1.9814	5.4420	.0241
39	2.3873	1.6202	8.5587	0.0000
40	4.7162	1.7165	3.4950	.0034
41	1.0458	1.0870	7.5507	0.0000
42	1.5686	2.7382	1.9814	0.0000
43	1.6477	2.0089	6.7630	0.0000
44	.6777	1.3313	3.2817	0.0000
45	.9735	1.2900	2.4355	0.0000
46	.0585	1.0114	3.3196	0.0000
47	.7258	.6433	1.2865	0.0000
48	.3715	1.0939	3.2680	0.0000
49	.3165	.5091	.9322	0.0000
50	.3371	.4369	1.9229	0.0000
51	.2442	.6880	.9735	0.0000
52	.1514	.2270	1.3588	0.0000
53	.1376	.4919	.2821	0.0000
54	.0688	.1995	1.1352	0.0000
55	.0310	.2236	.1823	0.0000
56	.0516	.3027	.2030	0.0000
57	.0206	.1582	.7190	0.0000
58	.0241	.2649	0.0000	0.0000
59	.0241	.1995	.5745	0.0000
60	.0172	.0998	.1204	0.0000
61	0.0000	.0310	.1479	0.0000
62	.0241	.1101	.3818	0.0000
63	0.0000	.1066	.0929	0.0000
64	.0034	.1101	.2580	0.0000
65	.0034	.1238	.0482	0.0000
66	.0034	.0516	.0138	0.0000
67	.0034	.0069	.1720	0.0000
68	0.0000	.0482	.0206	0.0000
69	0.0000	.1032	.0103	0.0000
70	0.0000	0.0000	.0757	0.0000
71	0.0000	.0206	.0275	0.0000
72	0.0000	.0413	.0103	0.0000
73	0.0000	0.0000	.0310	0.0000
74	0.0000	.0034	.0034	0.0000
75	0.0000	.0103	.0103	0.0000
76	0.0000	.0034	.0138	0.0000
77	0.0000	.0034	.0034	0.0000
78	0.0000	0.0000	.0069	0.0000

Figure 3.3-2. Probability of Occurrence of Intensity Levels of Each Band of Scene 28 (Spacecraft Data)

PERCENT OCCOURANCE OF INTENSITY LEVELS

LEVEL	BAND1	BAND2	BAND3	BAND4
0	0.0000	0.0000	0.0000	0.0000
1	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000	0.0000
4	0.0000	0.0000	0.0000	.0482
5	0.0000	0.0000	0.0000	.0413
6	0.0000	0.0000	0.0000	.0172
7	0.0000	0.0000	0.0000	.0103
8	0.0000	0.0000	0.0000	.0138
9	0.0000	0.0000	0.0000	.0138
10	0.0000	0.0000	0.0000	0.0000
11	0.0000	0.0000	0.0000	.0310
12	0.0000	0.0000	.0172	.0722
13	0.0000	0.0000	.0413	.2202
14	0.0000	.0069	.0206	.9460
15	0.0000	0.0000	.0069	1.9917
16	0.0000	.0275	.0103	6.3330
17	0.0000	.0344	.0103	9.6732
18	0.0000	.2339	.0069	11.7165
19	0.0000	1.1042	.0034	13.0478
20	0.0000	2.4149	.0069	12.8552
21	0.0000	2.3805	.0103	12.1465
22	.0034	3.1957	.0034	9.9140
23	.0791	5.2632	.0103	6.4706
24	.3474	9.9518	.0241	5.4008
25	2.4802	8.0151	.0206	3.0100
26	3.4193	8.1424	.0585	1.9229
27	5.3079	4.3860	.2546	1.1799
28	5.4420	5.1531	.4403	.7671
29	13.8356	6.7079	1.1627	.6674
30	9.7110	5.3182	2.8689	.4334
31	9.3430	3.3127	3.7702	.2614
32	11.3897	3.6739	3.0306	.2718
33	2.4871	1.0079	6.1266	.2098
34	6.6047	3.0891	3.9422	.1376
35	2.7313	.6123	8.2491	.0791
36	2.7864	3.4709	4.8504	.0516
37	4.2174	.6846	7.9257	.0172
38	5.3388	2.5800	6.6254	.0241
39	1.2281	1.3794	7.3753	0.0000
40	4.5855	1.2728	4.3619	.0034
41	1.1765	3.0478	6.7458	0.0000
42	2.1844	2.0777	2.4389	0.0000
43	1.0320	1.5033	6.7079	0.0000
44	.3956	1.6305	2.1053	0.0000
45	1.3141	.8290	3.8803	0.0000
46	.1651	1.2418	1.7475	0.0000
47	.5710	.3199	2.7417	0.0000
48	.3612	1.4310	1.9298	0.0000
49	.3818	.2339	2.0674	0.0000
50	.3956	1.0182	.9701	0.0000
51	0.0000	.1754	1.8129	0.0000
52	.3062	.6261	.6777	0.0000
53	.0757	.0894	.9506	0.0000
54	.1238	.4644	.7602	0.0000
55	.0241	.0894	.3474	0.0000
56	.0550	.2890	.3681	0.0000
57	0.0000	.3096	.3406	0.0000
58	.0241	.1926	.3681	0.0000
59	.0275	.0654	.2500	0.0000
60	.0069	.0860	.3543	0.0000
61	.0206	.0791	.1720	0.0000
62	0.0000	.2270	.1032	0.0000
63	.0069	.0310	.3715	0.0000
64	.0069	.1892	.0654	0.0000
65	.0034	.0241	.1617	0.0000
66	0.0000	.0275	.0757	0.0000
67	0.0000	.0998	.0482	0.0000
68	0.0000	.0447	.0826	0.0000
69	.0034	.0034	.0206	0.0000
70	0.0000	.0722	.0206	0.0000
71	0.0000	.0103	.0722	0.0000
72	0.0000	.0069	.0138	0.0000
73	0.0000	.0172	.0206	0.0000
74	0.0000	.0069	.0103	0.0000
75	0.0000	0.0000	.0172	0.0000
76	0.0000	.0103	0.0000	0.0000
77	0.0000	0.0000	.0103	0.0000
78	0.0000	.0034	.0069	0.0000

Figure 3.3-3. Probability of Occurrence of Intensity Levels in Each Band of Scene 28 (Ground-Processed Data)

```
127127 94 36 23 29 20 16 16 16 16 19 20 19 17 16 16 16 14 14 14 16 16 14
127127 86 35 18 21 15 12 15 15 16 16 18 24 20 15 15 16 14 15 15 15 16 15
 80 57 26 21 20 15 14 12 11 12 12 12 12 15 15 14 15 18 16 16 15 15 16 18
 10 14 21 20 12 12  7  1  0  0  0  0  3  3  1  3  4  6  4  4  6  7  6  6
 49 51 65 83 65 41 20 14 13 12 12 13 13 15 15 15 15 15 15 15 16 16 18 18
 65 69105124102 69 37 19 12 12 12 12 14 14 15 15 16 16 16 18 16 16 18 16
 53 72106127127127100 49 13 12 12 11 12 13 13 14 14 14 14 17 16 14 16 16
 68103127127127127113 62 22 14 11 13 11 11 13 14 15 15 16 15 15 15 16 15
 79116127127127127127112 67 28 13 11 11 11 13 14 15 16 16 15 15 15 15 16
127127127127127127127127 73 16  0  0  0  0  0  1  3  3  3  1  3  3  5  7
 92120123125120125127104 57 21 14 13 13 14 14 15 16 15 15 15 15 15 15 16
 45 69 81105 99109118 84 37 18 15 13 13 13 14 15 15 14 15 15 15 14 15 15
 17 39 81 96 94 96100 87 41 16 16 17 14 12 13 13 13 14 13 13 13 14 14 13
 15 37 70 86 93 97 77 35 16 16 18 16 16 15 14 13 14 14 13 13 14 14 16 16
 11 20 31 36 28 21 18 15 14 15 20 18 21 18 14 13 14 14 14 14 15 14 15 18
  1  7  3  1  3  1  0  5 18 53 69 34 13  7  5  3  3  3  1  3  3  1  3 13
 14 13 14 16 16 16 16 23 51 63 74 71 57 58 47 20 15 16 15 15 14 15 15 21
 15 14 15 16 16 16 15 21 37 49 77 90 99 94 58 23 18 16 15 16 16 15 15 16
 14 14 14 16 14 16 16 14 16 23 49 77 84 77 81 69 65 57 29 17 17 16 14 14
 14 14 14 15 14 15 14 14 15 27 59 70 52 50 73 97110113 97 62 27 18 16 16
 15 15 15 15 14 15 15 14 19 43 74 74 61 77105127127125 96 65 28 16 15 15
  4  4  5  4  4  5  8  4  8 23 59 65 65 76100115127127 72 27 15  9  5  4
 15 14 14 14 15 15 15 15 15 19 28 33 33 31 47 89107 92 62 33 21 23 18 14
 15 14 14 14 15 14 15 15 14 15 15 18 16 14 21 45 68 68 33 19 25 31 19 15
 14 13 14 16 14 13 14 14 14 14 14 16 13 11 12 23 36 46 36 22 20 24 30 26
 14 15 15 15 14 14 14 15 14 14 14 13 11 10 10 13 15 22 24 21 18 20 27 24
 16 14 14 14 14 14 14 15 15 15 14 14 13 13 14 13 13 15 16 15 15 22 25 16
  5  5  5  5  4  4  5  4  5  4  4  5  4  8  5  4  4  4  5  4  5  8  9  9
 30 28 16 15 15 16 15 16 16 15 15 14 15 18 15 15 15 15 15 16 15 16 15 16
 63 35 15 14 14 14 14 16 15 15 15 15 15 15 14 14 14 15 16 15 16 15 14 15
 96 44 17 14 12 12 13 14 14 17 17 14 14 16 16 13 13 13 14 14 14 14 14 13
 66 44 27 20 13 10 12 10 12 13 13 14 15 14 15 14 13 14 14 14 15 15 15 15
 36 28 25 16 12 13 12 10 12  9 10 12 13 15 15 15 15 15 15 15 15 14 15 15
 13 16 21 13  4  3  3  1  1  1  1  3  3  4  5  5  7  7  7  7  5  5  9  9
 27 44 64 51 23 12 12 12 11 11 12 12 12 12 13 12 13 15 15 16 16 15 14 14
 15 15 12 12 10 10 10 10 10  9  9 10 10 10 11 10 10 10 11 12 15 14 11 10
 11 11 11 10 10 10 10 11 12 10 10 10 10 11 11 12 12 10 11 11 11 10 10 10
 10 10 10 10 10  9  9 10 10 10 10  9 10  9 10 10 10 10 12 13 10  9 10 10
 10  9 10 10 10 10 12 10 10 12 10 10  9 10  9 12 13 13 13 14 13 10  9 10
  4  1  1  1  1  1  1  1  0  1  3  3  1  0  3  4  4  7 11  9  3  1  1  1
 16 14 13 13 11 11 11 11 11 11 12 13 11 12 14 16 18 16 16 16 15 13 15 15
 18 18 15 15 14 12 11 12 11 11 12 15 14 14 15 16 21 25 31 51 57 59 71 57
```

Figure 3.3-4. A Segment of Input Data from Scene 8 (Band 2)

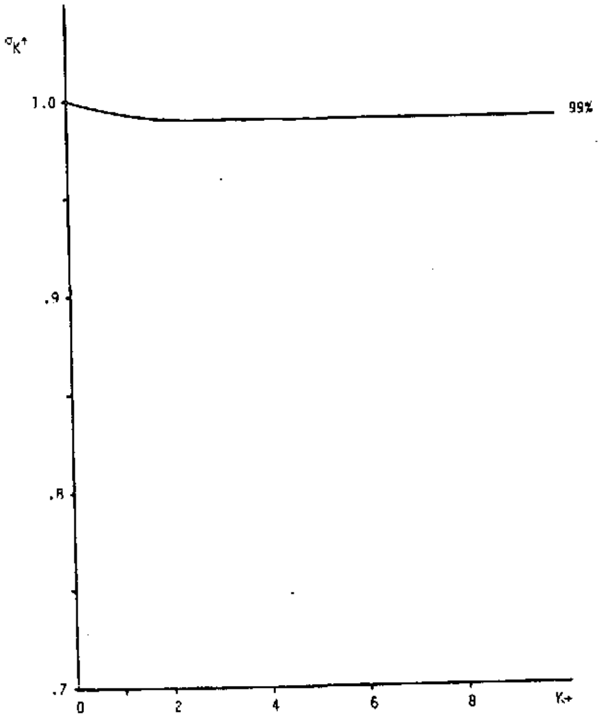


Figure 3.3-5. Cross Spectral-Spacial Correlation of Scene (Ocean)

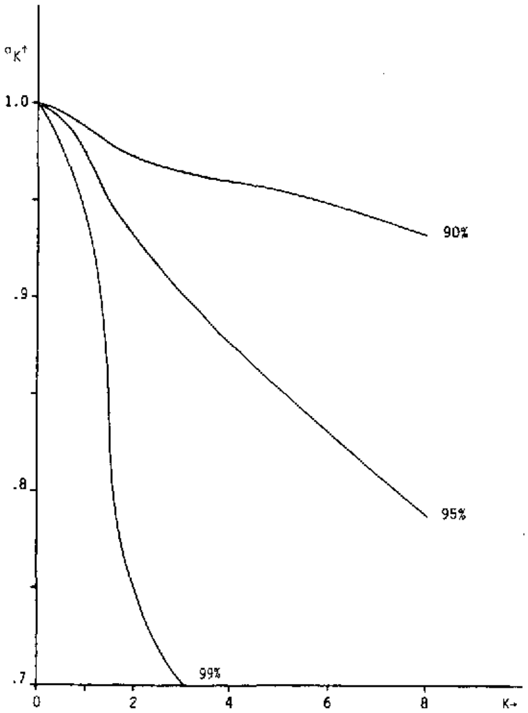


Figure 3.3-6. Cross Spectral-Spacial Correlation of Scene (Forest)

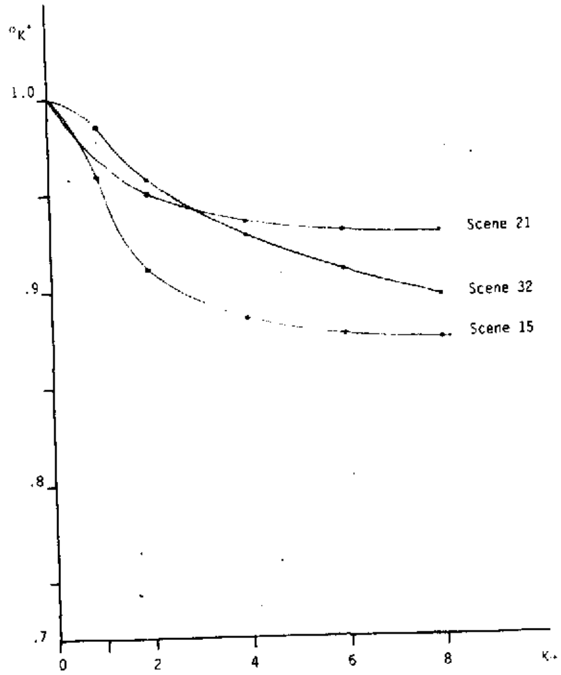


Figure 3.3-7. Spectral-Spacial Correlations of Scenes 15, 21, and 32 (95% Curves Shown)

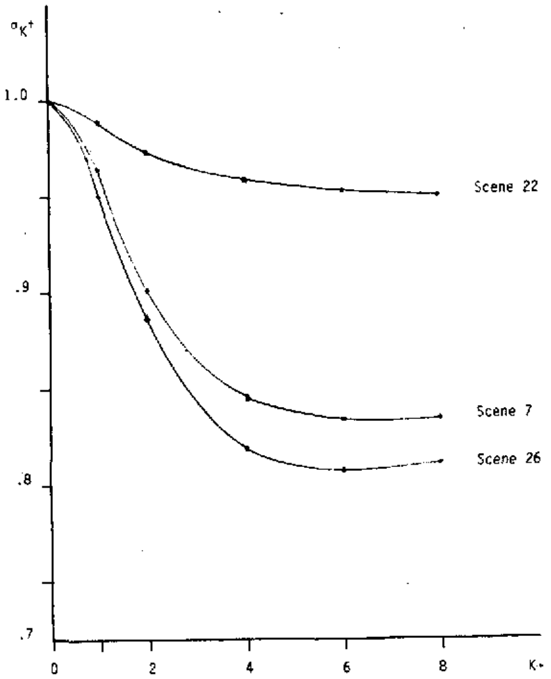


Figure 3.3-8. Spectral-Spacial Correlations of Scenes 7, 22, and 26 (95% Curves Shown)

```
    -16-15-14-13-12-11-10 -9 -8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
-16
-15
-14
-13
-12
-11
-10
 -9
 -8
 -7
 -6
 -5
 -4
 -3
 -2                                                   3
 -1                                               11 37 11
  0                                               30100 30
  1                                               11 37 11
  2                                                   2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 16
```

Figure 3.3-9. Joint Probability Density of Band 3 and Band 4, Scene 4

	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-16																1	1																
-15																	1																
-14																1	1																
-13																2	1																
-12																1	2																
-11																2	3																
-10																2	3	1	1		1		1										
-9															1	2	3	2	1	2	1	1											
-8														2	2	4	4	5	3	3	2	1											
-7														1	3	3	5	5	4	3	2	2											
-6										1		1	1	3	5	7	9	7	8	5	3	2	1										
-5													2	3	5	6	10	7	7	4	4	2	1										
-4											2	3	4	6	9	17	22	18	15	11	7	5	2	1									
-3											1	1	3	4	6	10	16	12	9	6	4	3	2	1									
-2								2	2	3	3	6	8	13	22	34	44	38	27	15	10	4	4	2	1								
-1											1	1	2	5	7	16	29	17	9	5	4	3	2	2	1	2							
0						1	1	3	3	4	6	9	11	25	38	69	100	64	37	24	15	8	5	4	3	2	2	1					
1											1	2	3	5	9	16	26	14	8	4	3	2	2	1	1								
2					1	1		2	2	3	4	5	11	18	25	36	43	30	17	15	6	4	3										
3										1	2	2	3	6	10	10	13	8	6	4	2	2	1										
4									1	2	2	4	8	11	18	18	20	13	9	6	2												
5											1	2	3	6	7	8	9	7	4	3	2	1											
6											2	3	3	6	9	8	8	6	4	2	2												
7											1	1	1	3	5	4	5	4	2	2	1												
8												1	2	2	3	4	4	3	2	2	2												
9												1	2	2	2	2	2	3	1														
10														1	2	2	2	2	2														
11														1	1	2	2																
12																	2	1															
13																1	1																
14																	1																
15																																	
16																																	

Figure 3.3-10. Joint Probability Density of Band 3 and Band 4, Scene 23

	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-16																																	
-15																																	
-14																																	
-13																																	
-12																																	
-11																																	
-10																																	
-9																	1	1															
-8															1	2	2	3	1		1												
-7															2	2	4	4	3	2	1												
-6														1	2	4	7	6	4	3	2	1											
-5														2	4	6	8	6	5	3	2	1	1										
-4													2	4	9	15	19	16	13	7	4	2	2										
-3													1	3	5	9	12	9	6	4	3	1											
-2											1	2	5	9	20	33	41	36	22	14	6	4	2										
-1														2	4	10	21	12	7	3	2	1											
0										2	3	6	8	19	36	67	100	66	35	19	11	5	2	1									
1													1	3	5	12	20	12	5	2	2												
2										1	2	3	6	14	24	40	41	35	18	10	5	2	1										
3												2	2	5	6	9	11	9	4	3	1	1											
4											2	3	4	6	11	16	17	13	8	4	2												
5												2	3	4	5	6	7	6	3	2													
6												1	2	3	5	5	6	5	3	2													
7														2	3	3	4	2	2														
8													1	1	2	2	2	2	1														
9														1	1	2	1	1															
10																1	1																
11																																	
12																																	
13																																	
14																																	
15																																	
16																																	

Figure 3.3-11. Joint Probability Density of Band 3 and Band 4, Scene 29

An estimate of the total system noise energy was made based on three uniform subscenes. While the figures are necessarily an upper bound on the sum of sensor, quantization, and de-compression noises, they serve to approximate the minimum data activity produced the the ERTS system. Table 3.2 lists the variances obtained for each spectral band based on the selected subscenes:

Table 3.2. Estimated System Noise Energy

	Scene 1	Scene 3	Scene 4
Band 1	.326	.964	.924
Band 2	.343	.576	.780
Band 3	1.350	.666	.579
Band 4	.844	.304	.332

The noise component produced by the scanner mechanism is a function of light intensity and scenes 3 and 4 are based on water which produces low incident light levels whereas scene 1 is based on cloud cover which entails a high intensity level. No estimates were taken for scenes of intermediate intensity since the data activity in such areas is much greater than the system noise level.

3.4 DATA COMPRESSION CHARACTERISTICS AND STATISTICS

3.4.1 Significance of the Compression Statistics Measured

For each subscene and full scene processed, several key statistics and global measures of the data compression performance were obtained. These measurements serve as an indicator of the compression to be expected on similar subscenes containing the same object class and permit an evaluation of the efficacy of each compression technique for similar data.

The probability distribution of the symbols obtained by each algorithm indicates the level of performance to be expected when using that algorithm on the data. The greater the clustering of these symbols about zero, the

the higher the compression when the symbols are encoded for the Rice or Huffman techniques. For most data, the variance of the data decreases progressing from the SSDI to the SSDIA and from the SSDIA to the SSDIAM algorithms. For anomalous data, such as that obtained with a defective sensor, this may not be the case. The entropy of the symbol distribution is also computed for each algorithm since the entropy forms a lower bound to the average compressed bit rate.

Based on this symbol distribution, the computer generates the Huffman code for each technique. Since the symbol statistics are measured globally for the entire scene, this code would be used for non-adaptively encoding that scene. The grouped Huffman code, as described in Section 2.2.3 is used in order to simulate a technique which has shown high promise for ground data compression usage. The codeword assigned to each symbol is printed together with the grouped codeword prefix beside those symbols which form the grouping. In addition, the number of bits in the codeword for each symbol, the total probability of symbols in the grouping, and the average compressed bit rate for the scene are displayed. The Huffman coding efficiency for each technique can be obtained by comparing the average bit rate with the symbol entropy. In addition to obtaining the global Huffman average bit rate for the scene based on each technique, the average bit rate is computed for the scene based on use of the adaptive Huffman or Rice algorithms for the encoding of either the SSDI or SSDIA symbols. Since the adaptive Huffman technique generates a new Huffman code for each block of data, no output of these codewords is practical.

The percentage occurrence of the three Rice modes (FS, CFS, $\overline{\text{CFS}}$) are printed as are the percentage occurrence of the split-pixel modes. The percentage of Rice modes which occur are dependent on the overall data activity within the scene as well as on local variations of data activity within the scene. Thus, even a scene which has an overall high data activity normally has some subregions with moderate or low activity. The split-pixel modes only occur when the data activity in a block produces an average bit rate greater than 4 bits/sample.

The time-varying data compression of the scene is also printed for the global Huffman, adaptive Huffman, and Rice codes as the average bit rate per scan line. These varying bit rates fluctuate corresponding to the average data activity in each scan line. Trends of data activity and the effects of sensor or system anomalies can be discerned by observing these statistics. In addition, the performance of the three techniques are compared on a line-by-line basis as well as by judging the overall compressed bit rate of each technique. While one of the three techniques will produce the lowest bit rate on most scan lines of a given scene there will be some lines in which a lower rate is produced by a competing technique.

The buffer statistics are computed for the selected SSDI mode with Rice encoding for the two fixed-rate buffer outputs of 3.0 and 3.5 bits per sample. These rates were selected since they correspond to an average fixed-rate compression of 2:1 or more and they bound intermediate buffer output rates. The values printed for each scan line represent the total accumulated number of bits in the buffer at the end of the scan line, assuming zero buffer bits at the start of the first scan line. The buffer total increases when the average input bit rate for the scan line surpasses the fixed output rate and the total decreases when the average input bit rate is less than the output bit rate. Buffer underflow is printed as zero bits. These buffer statistics are primarily a consideration for compression performed aboard a spacecraft where the tradeoff of transmitted data rate versus buffer capacity must be considered.

3.4.2 Results Obtained

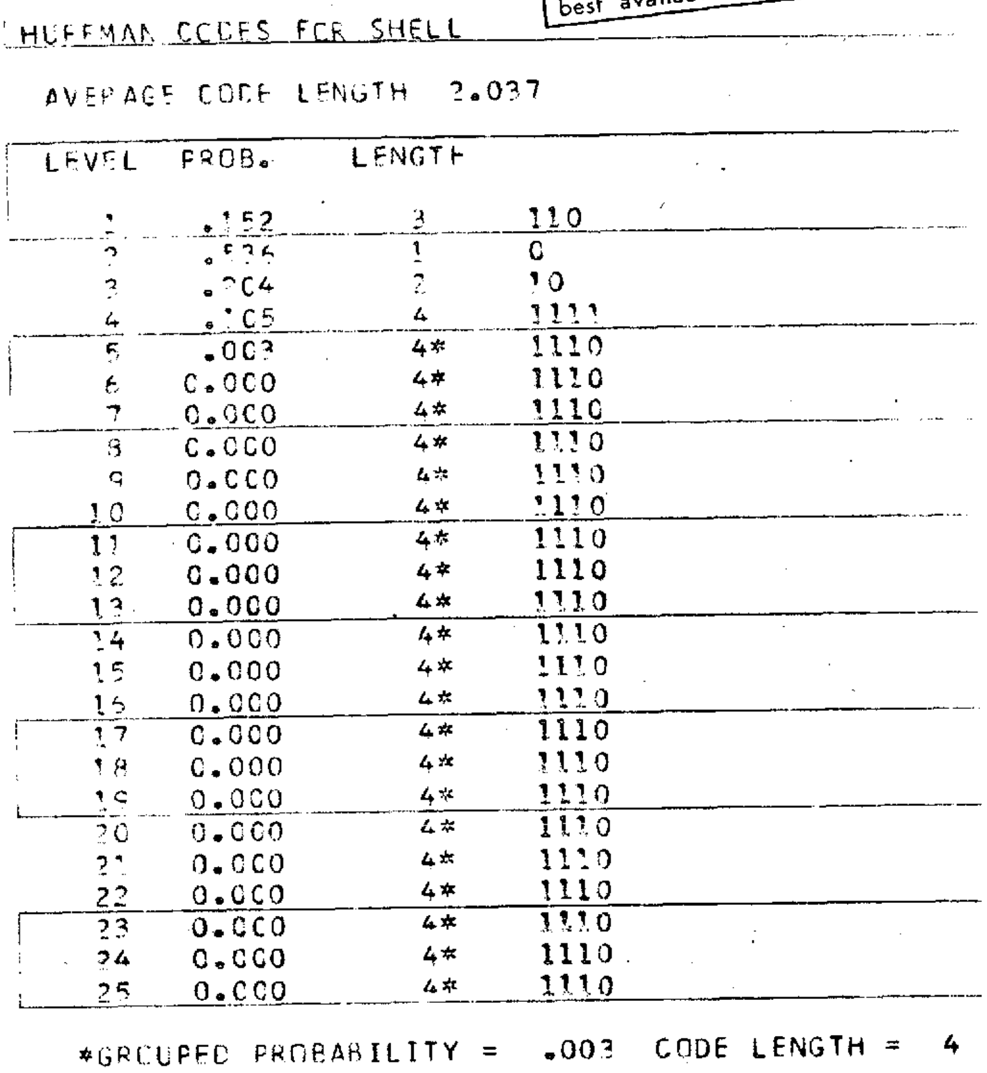
The overall data compression achieved on each scene processed is summarized in Table 3.3 for all the compression techniques used. The first column contains the scene identification number, correlated with Table 3.1. Columns six through nine give the average bit rate achieved on the scene by the Shell, SSDI, SSDIA, and SSDIAM (shown for single-level mapping, $|m| = 1$) algorithms followed by global Huffman coding. Columns two and three give the average bit rate for the scene as produced by the adaptive Huffman and Rice algorithms for symbols generated by either the SSDI or SSDIA (see column 5). Column 4 gives the peak buffer fill (in bits) generated by the scene for an output buffer rate of 3.5 bits per input intensity sample. Table 3.3 permits the determination of the expected bit rate that can be achieved on the various object classes using the different algorithms. The bit rate produced varies from a low of 1.220 bits to a high of 3.821 bits.

Table 3.3. Compressed Bit Rates of Scenes Processed

	ADAPTIVE ALGORITHMS			GLOBAL HUFFMAN				
SCENE NUMBER	HUFFMAN	RICE	PEAK BUFFER FILL (BITS)	SYMBOLS	SHELL	SSDI	SSDIA	SSDIAM
1	1.42	1.31	0	SSDI	1.697	1.508	1.491	1.427
2	1.45	1.22	0	SSDIA	1.912	1.920	1.458	1.380
3	1.49	1.27	0	SSDIA	2.037	2.123	1.532	1.485
4	1.53	1.37	0	SSDIA	2.162	1.953	1.618	1.461
5	1.59	1.73	0	SSDIA	1.711	1.687	1.934	1.887
6	1.78	1.79	0	SSDIA	2.014	2.135	1.890	1.453
7	2.39	2.41	0	SSDIA	2.386	2.377	2.478	2.182
8	2.46	2.66	28,493	SSDI	2.842	2.784	3.089	2.648
9	2.47	2.48	0	SSDIA	2.945	2.750	2.521	2.069
10	2.61	2.58	0	SSDIA	3.043	3.156	2.762	2.285
11	2.55	2.67	22,328	SSDI	2.827	2.790	3.251	2.724
12	2.64	2.67	1,134	SSDIA	2.956	3.106	2.861	2.559
13	2.71	2.85	141	SSDIA	3.160	2.883	3.106	2.693
14	2.81	2.76	0	SSDIA	3.233	3.268	3.096	2.655
15	2.89	2.48	19,415	SSDIA	3.135	3.374	2.264	2.767
16	2.91	2.99	1,227	SSDIA	2.915	2.759	3.351	2.956
17	3.01	2.98	0	SSDIA	3.322	3.399	3.129	2.522
18	3.07	3.02	2,411	SSDIA	3.206	3.097	3.167	2.558
19	3.11	3.04	0	SSDIA	3.438	3.447	3.185	2.541
20	3.12	3.21	1,862	SSDI	3.436	3.346	3.556	2.939
21	3.12	3.19	9,801	SSDI	3.349	3.435	3.433	3.064
22	3.18	3.21	3,053	SSDI	3.374	3.394	3.408	2.877
23	3.26	3.28	1,213	SSDIA	3.644	3.467	3.452	3.034
24	3.32	3.35	3,670	SSDI	3.581	3.536	3.487	2.981
25	3.36	3.38	88,775	SSDIA	3.407	3.283	3.667	3.272
26	3.37	3.35	3,091	SSDI	3.526	3.650	3.355	2.824
27	3.39	3.40	3,355	SSDIA	3.816	3.699	3.559	2.935
28	3.45	3.44	5,734	SSDI	3.654	3.698	3.533	2.907
29	3.48	3.47	3,872	SSDIA	3.512	3.479	3.651	3.064
30	3.56	3.54	9,815	SSDIA	3.757	3.747	3.660	2.981
31	3.47	3.59	168,599	SSDI	3.739	3.686	3.676	3.046
32	2.93	3.00	6,649	SSDIA	3.239	3.339	3.323	2.966
33	3.52	3.53	168,450	SSDIA	3.715	3.821	3.633	3.025
34	2.68	3.04	0	SSDI	3.091	3.044	3.261	2.827

In general, the Shell and SSDI algorithms gave comparable compressed bit rates as did the adaptive Huffman and Rice techniques. For all scenes the SSDIAM produced a lower bit rate than the SSDIA. The difference in bit rates produced between the SSDI and SSDIA algorithm varies. For well-behaved data the SSDIA produces a lower bit rate than the SSDI but the reverse occurs for data in which the band 2 sensor generates anomalous data. This situation results from the averaging operation performed which includes intensities from a scan line of correct data and intensities from the scan line containing bad data. This produces large first differences within spectral band 2 and second differences which are not correlated with either band 1 or band 3. This effect produces disturbances in the SSDIA extending over 2 scan lines of compressed data. Each time a defective scan line occurs resulting in poor compression for one third of the scene. This averaging operation is also performed for the SSDIAM to produce a high compressed bit rate for such anomalous data. This effect is evident from the time-varying compressed bit rate shown in Figure 3.4-1.

The effect of such data anomalies is less severe for the SSDI algorithm but some degree of degradation is still produced with the severity depending on the form of source coding which is used for the SSDI symbol. The global Huffman code becomes less efficient because one sixth of the scan lines (those containing anomalous data) produce symbol statistics quite different than the statistics for the other lines. The Huffman code generated based on the symbol statistics for the entire scene is neither optimal for the normal data nor for the anomalous data symbols. This same variation in line symbol statistics corrupts one third of the scan lines when using the adaptive Huffman code which uses the statistics developed for the symbols on one scan line for encoding symbols from the next line of data, a process which requires a fairly high correlation of symbol statistics from line-to-line to be effective. The Rice technique normally performs better than the global or adaptive Huffman methods for such data since it generates a code based solely on the statistics of a block of symbols contained within a single scan line. This ability to rapidly adjust to changing statistics is advantageous for segments of defective data.



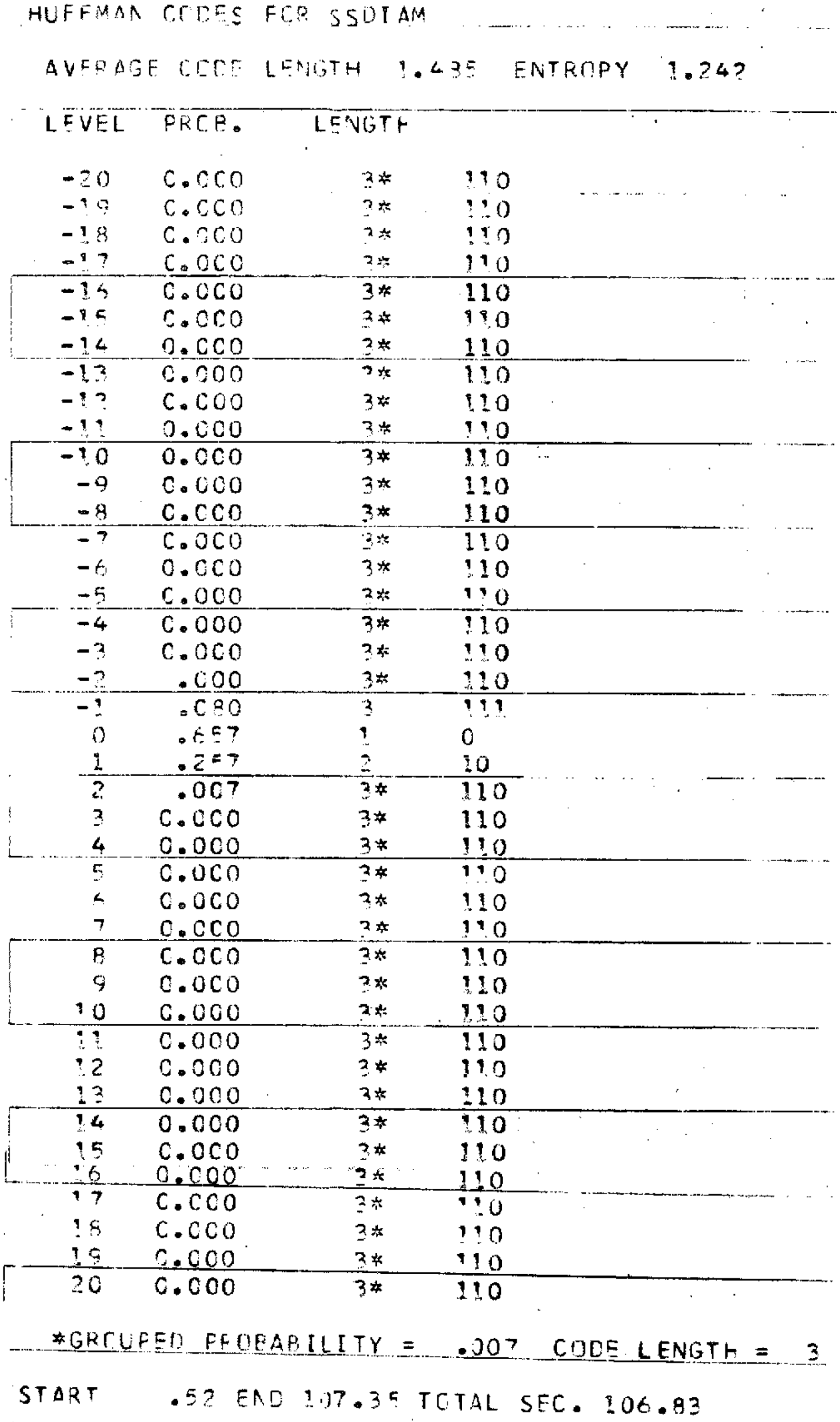
```
HUFFMAN CODES FOR SHELL

 AVERAGE CODE LENGTH  2.037

 LEVEL   PROB.     LENGTH

   1     .152        3       110
   2     .536        1       0
   3     .204        2       10
   4     .105        4       1111
   5     .003        4*      1110
   6    0.000        4*      1110
   7    0.000        4*      1110
   8    0.000        4*      1110
   9    0.000        4*      1110
  10    0.000        4*      1110
  11    0.000        4*      1110
  12    0.000        4*      1110
  13    0.000        4*      1110
  14    0.000        4*      1110
  15    0.000        4*      1110
  16    0.000        4*      1110
  17    0.000        4*      1110
  18    0.000        4*      1110
  19    0.000        4*      1110
  20    0.000        4*      1110
  21    0.000        4*      1110
  22    0.000        4*      1110
  23    0.000        4*      1110
  24    0.000        4*      1110
  25    0.000        4*      1110

   *GROUPED PROBABILITY =   .003   CODE LENGTH =   4
```

Figure 3.4-1: Global Huffman Code for Shell Symbols, Scene 3

Appendix B contains a complete set of computer output for the principal full scene, and contains the global Huffman code generated for the Shell, SSDI, SSDIA, and SSDIAM symbols. Figures 3.4-1 through 3.4-3 present global Huffman codes which illustrate typical forms such codes can assume. Figure 3.4-1 shows the Huffman code generated coding for the Shell symbols of scene 3, where shell 1 corresponds to the inner shell with maximum level of zero. The χ^2-distribution peaks at the second shell, and no levels are occupied beyond the fifth.

```
HUFFMAN CODES FOR SSDI

 AVERAGE CODE LENGTH  3.306  ENTROPY  2.951

 LEVEL  PROB.    LENGTH

  -20    .001       4*    1100
  -19    .001       4*    1100
  -18    .001       4*    1100
  -17    .001       4*    1100
  -16    .002       4*    1100
  -15    .002       4*    1100
  -14    .001       4*    1100
  -13    .002       4*    1100
  -12    .002       4*    1100
  -11    .003       4*    1100
  -10    .003       4*    1100
   -9    .003       8     11111111
   -8    .005       8     11111110
   -7    .006       7     1111001
   -6    .007       7     1111000
   -5    .010       7     1111101
   -4    .012       6     111000
   -3    .018       6     111011
   -2    .046       5     11010
   -1    .144       3     100
    0    .463       1     0
    1    .138       3     101
    2    .043       5     11011
    3    .017       6     111010
    4    .011       6     111001
    5    .008       7     1111100
    6    .007       7     1111011
    7    .006       7     1111010
    8    .004       8     11111101
    9    .003       8     11111100
   10    .003       4*    1100
   11    .003       4*    1100
   12    .002       4*    1100
   13    .002       4*    1100
   14    .002       4*    1100
   15    .002       4*    1100
   16    .001       4*    1100
   17    .001       4*    1100
   18    .001       4*    1100
   19    .001       4*    1100
   20    .001       4*    1100

 *GROUPED PROBABILITY =   .047  CODE LENGTH =  4
```

Figure 3.4-2. Global Huffman Code for SSDI Symbols, Scene 12



```
HUFFMAN CODES FOR SSDIAM

 AVERAGE CODE LENGTH  1.435  ENTROPY  1.242

 LEVEL   PROB.     LENGTH

  -20   0.000        3*     110
  -19   0.000        3*     110
  -18   0.000        3*     110
  -17   0.000        3*     110
  -16   0.000        3*     110
  -15   0.000        3*     110
  -14   0.000        3*     110
  -13   0.000        3*     110
  -12   0.000        3*     110
  -11   0.000        3*     110
  -10   0.000        3*     110
   -9   0.000        3*     110
   -8   0.000        3*     110
   -7   0.000        3*     110
   -6   0.000        3*     110
   -5   0.000        3*     110
   -4   0.000        3*     110
   -3   0.000        3*     110
   -2    .000        3*     110
   -1    .080        3      111
    0    .657        1      0
    1    .257        2      10
    2    .007        3*     110
    3   0.000        3*     110
    4   0.000        3*     110
    5   0.000        3*     110
    6   0.000        3*     110
    7   0.000        3*     110
    8   0.000        3*     110
    9   0.000        3*     110
   10   0.000        3*     110
   11   0.000        3*     110
   12   0.000        3*     110
   13   0.000        3*     110
   14   0.000        3*     110
   15   0.000        3*     110
   16   0.000        3*     110
   17   0.000        3*     110
   18   0.000        3*     110
   19   0.000        3*     110
   20   0.000        3*     110

 *GROUPED PROBABILITY =  .007  CODE LENGTH =  3

START     .52 END 107.35 TOTAL SEC. 106.83
```

Figure 3.4-3. Global Huffman Code for SSDIAM Symbols, Scene 3

Figure 3.4-2 presents a typical SSDI global Huffman code for a source symbol distribution based on data having a moderately high activity. The first column gives the symbol level, (only symbols between -20 and +20 are shown since symbols outside this range are always in the lumped group.) The second column gives the probability of occurrence of that symbol, the third column gives the number of bits in the symbol codeword, and the last column gives the actual codeword generated. Symbols having an asterisk by the length indicate that the symbol forms a part of the lumped grouping. In this example, the lumped codeword prefix is 1100, of length four bits. The total probability of symbols included in the lumped grouping is .033 for this example. The average codeword length for this scene is 3.106 bits/sample and the symbol entropy is 2.951 bits/sample.

The global Huffman code for the SSDIAM symbols generated for scene 3 is shown in Figure 3.4-3. This code is typical of those generated for data having a low source activity and only three symbols, of total probability is .994, are encoded directly with the remainder falling in the lumped group with prefix codeword 110. Since no Huffman codeword contains less than one bit, the ratio of the average codeword length to the symbol entropy increases as the entropy decreases. This is evident in Figures 3.4-2 and 3.4-3 where the respective ratios are 1.053 and 1.2.

Figures 3.4-4 and 3.4-5 contain segments of time-varying compression and buffer statistics for scenes of high and low data activity. These two figures convey a great deal of information regarding data characteristics and permit comparisons of the performance and limitations of the various algorithms. The first column of Figure 3.4-4 gives the scan line numbers (from the beginning of the subscene), columns 2 and 3 gives the buffer fill (in bits) at the end of each scan line based on fixed output rates of 3.0 and 3.5 bits/sample; columns 4 though 6 give the average bit rate for that line based on the global Huffman, adaptive Huffman, and the Rice algorithms respectively. For this Figure, SSDI symbols are used and the buffer statistics are based on the global Huffman output of scene 12.

Several observations can be made concerning Figure 3.4-4. First, the source data activity increases from line 36 to about line 61 and then begins to decrease. As the average bit rate per line for column 4 increases above the buffer output rates, the buffer fill increases until the rates of column 4 fall below the buffer rates. The peak buffer fill for an output rate of 3 bits occurs on line 67 while the peak buffer fill for an output rate of 3.5 bits occurs on line 62.

Scan Line	Buffer Bits		Average Bits/Sample		
Number	3.0	3.5	Global Huffman	Adaptive Huffman	Rice
36	0	0	1.990	2.150	1.854
37	0	0	1.851	1.815	1.640
38	0	0	2.195	2.231	2.003
39	0	0	2.019	2.167	2.007
40	0	0	1.975	1.954	1.866
41	0	0	2.257	2.068	2.118
42	0	0	1.731	1.710	1.427
43	0	0	1.879	1.729	1.738
44	0	0	1.982	1.906	1.728
45	0	0	1.768	1.729	1.628
46	0	0	1.805	1.650	1.688
47	0	0	1.970	1.769	2.024
48	0	0	1.751	1.817	1.353
49	0	0	2.152	1.970	2.122
50	0	0	2.606	2.484	2.446
51	0	0	2.421	2.443	2.336
52	0	0	2.522	2.548	2.798
53	0	0	2.997	2.979	3.442
54	0	0	2.921	3.440	3.310
55	130	0	3.193	3.461	3.302
56	561	95	3.641	3.814	4.125
57	1022	220	3.686	3.659	4.275
58	1404	266	3.568	3.943	3.790
59	1981	507	3.859	4.210	4.571
60	2443	633	3.688	3.982	4.253
61	2992	846	3.817	3.823	5.446
62	3538	1056	3.813	3.943	4.039
63	3862	1044	3.482	3.749	4.336
64	4288	1134	3.634	3.780	3.736
65	4041	551	2.632	2.710	2.304
66	4402	576	3.537	3.519	3.847
67	4501	339	3.147	3.345	3.262
68	4420	0	2.879	3.006	2.884
69	4208	0	2.685	2.863	2.655
70	3712	0	2.262	2.375	2.205
71	3255	0	2.320	2.342	2.158
72	2894	0	2.463	2.243	2.339
73	2142	0	1.881	1.996	1.692
74	1354	0	1.827	1.766	1.625
75	787	0	2.156	2.179	1.830
76	0	0	1.732	1.701	1.516
77	0	0	2.086	1.933	1.981

Figure 3.4-4. Time-Varying Compression Statistics, Scene 12

Scan Line Number	Buffer Bits $R_{OUT} = 3.0$	Global Huffman	Adaptive Huffman	Rice
36	0	1.347	1.381	1.179
37	0	1.564	1.494	1.644
38	0	1.634	1.661	1.676
39	0	1.580	1.580	1.350
40	0	1.315	1.443	1.257
41	0	1.582	1.582	1.250
42	0	1.376	1.408	1.246
43	0	1.586	1.509	1.673
44	0	1.606	1.606	1.652
45	0	1.452	1.452	1.092
46	0	1.347	1.427	1.244
47	0	1.586	1.586	1.237
48	0	1.285	1.405	1.259
49	0	1.771	1.683	1.845
50	0	1.329	1.344	1.305
51	0	1.436	1.406	1.049
52	0	1.510	1.808	1.600
53	0	1.467	1.467	1.054
54	0	1.322	1.375	1.158
55	0	1.739	1.315	1.708
56	0	1.351	1.506	1.423
57	0	1.417	1.417	1.046
58	0	1.445	1.530	1.467
59	0	1.475	1.448	1.088
60	0	1.342	1.405	1.223
61	0	1.789	1.717	1.749
62	0	1.501	1.756	1.716
63	0	1.432	1.552	1.182
64	0	1.281	1.238	.856
65	0	1.382	1.265	.945
66	0	1.329	1.384	1.205
67	0	1.749	1.689	1.753
68	0	1.488	1.438	1.613
69	0	1.376	1.469	1.106
70	0	1.330	1.265	.912
71	0	1.381	1.266	.958
72	0	1.320	1.366	1.165
73	0	1.760	1.698	1.693
74	0	1.454	1.662	1.615
75	0	1.348	1.348	1.040
76	0	1.411	1.296	1.003
77	0	1.445	1.415	1.243

Figure 3.4-5. Time-Varying Compression Statistics, Scene 3

On lines of lower data activity the Rice algorithm normally yields the lowest average bit rate. One reason for this is that the Rice algorithm can produce a bit rate as low as .37 bits/sample while the Huffman codes can never be less than 1 bit/sample. As the average bit rate increases the Huffman codes can perform better than the Rice in some cases since they do not require the transmission of overhead bits for each block of data.

Another limitation of the Rice algorithm, as simulated here, is the use of fixed codes for the fundamental sequence and these fixed codes are not optimal for all blocks. The performance of the global Huffman code depends on how well the symbol statistics of a scan line match the symbol statistics developed for the entire scene and the performance of the adaptive Huffman code is dependent on the correspond of the symbol statistics of the line being encoded with the symbol statistics of the preceding line. Therefore, while the adaptive algorithms yield performance superior to the fixed Huffman algorithm for the entire scene, this is not necessarily true on the basis of any individual scan line. As an example, the global Huffman is best for line 52, the adaptive Huffman is best for line 41, and the Rice technique is best for line 36.

Figure 3.4-5 portrays similar time-varying statistics based on SSDIA symbols for a segment of data from scene 3 having low source activity. This scene also differs from the preceding in that the data contains anomalous sensor output on every sixth scan line. The effect of this sensor defect disturbs the various algorithms in differing degrees as can be observed in the performance shown in the figure. Since the Rice algorithm encodes each line separately, the increased bit rates on lines 37, 38, 43, 44, etc., reflect the disturbance of the SSDIA symbols themselves. Since the adaptive Huffman algorithm uses the statistics of symbols in one line for encoding the next line, the anomalous data produces effects that propogate further. The global Huffman is an intermediate case since each line is encoded based on the code developed for the entire scene. This global code is affected by the defective sensor data statistics and produces a code which generates a higher bit rate for all scan lines. Figure 3.4-5 also shows lines for which the Rice algorithm produces average bit rates of less than one bit/sample including overhead bits.

Figures 3.4-6 through 3.4-9 show buffer statistics for two scenes based on the SSDIA/Rice algorithm. Figures 3.4-6 and 3.4-7 are based on scene 15 which is centered on Catalina Island and extends into the Pacific ocean on either side. This scene produces an average bit rate of 2.89 bits. Figure 3.4-6, based on a buffer output rate of 3.5 bits, shows buffer underflow until scan line 48 , at which point the average compressed bit rate exceeds 3,5 bits per line. The bit rate per line begins to fall below the buffer output rate around scan line 107 at which point the buffer fill decreases.

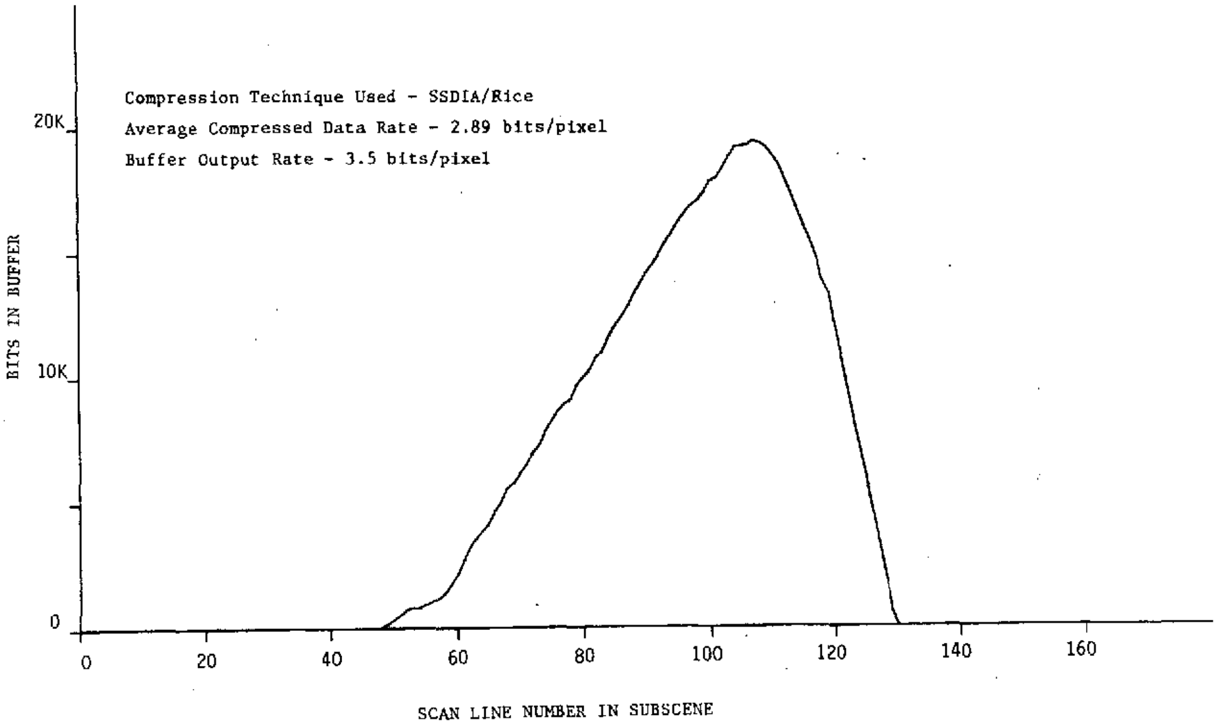


Figure 3.4-6. Buffer Statistics of Scene 15 (Catalina Island)

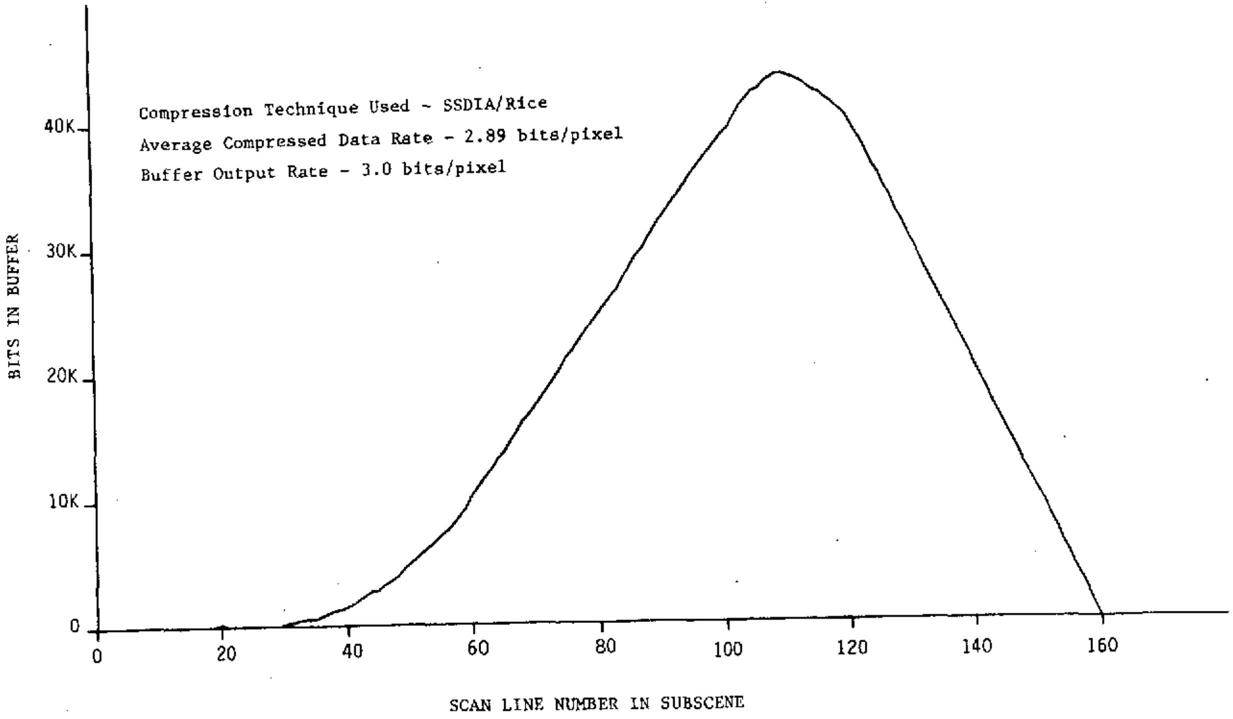


Figure 3.4-7. Buffer Statistics of Scene 15 (Catalina Island)

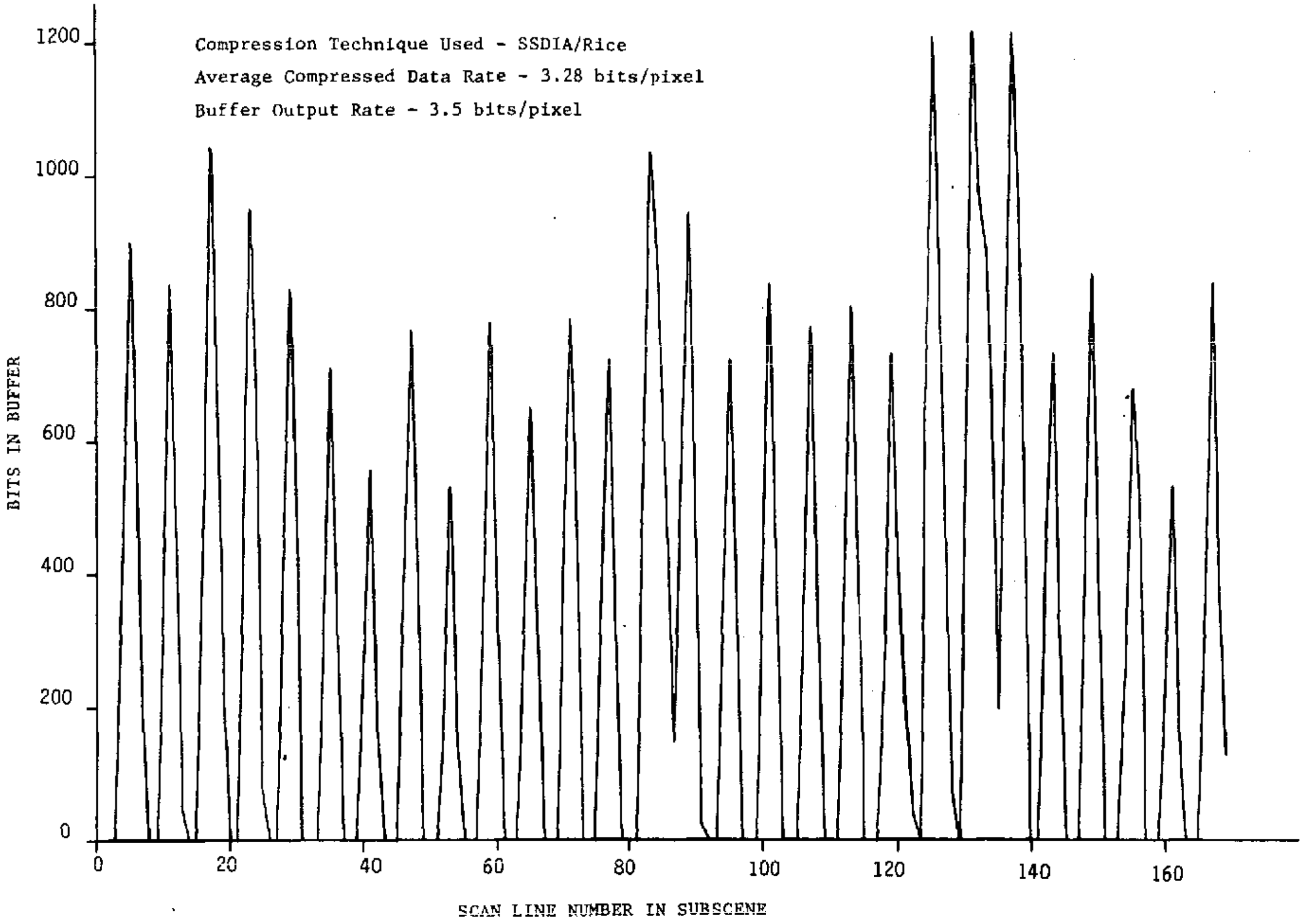


Figure 3.4-8. Buffer Statistics of Scene 23 (Saguenay River)

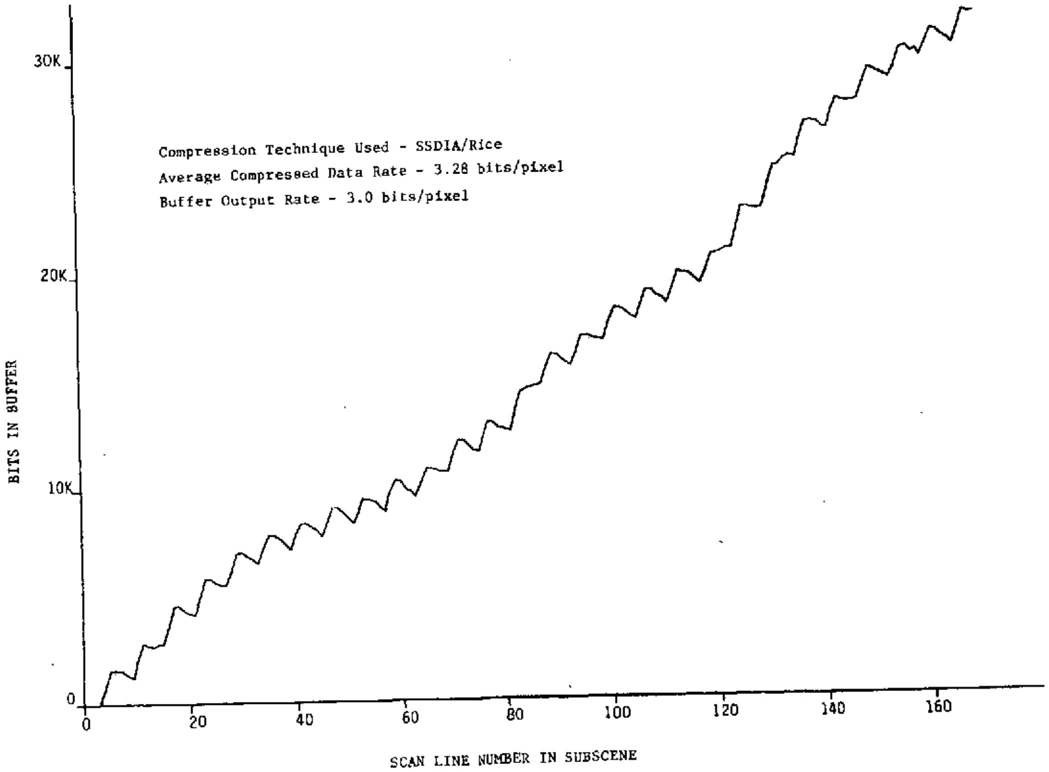


Figure 3.4-9. Buffer Statistics of Scene 23 (Saguenay River)

Figure 3.4-7, based on a buffer output rate of 3 bits, illustrates the same effect except that buffer fill begins around scan line 29 where the compressed bit rate exceeds 3 bits/line and peaks at scan line 110. For an output buffer rate of 3 bits the peak buffer fill is 43,632 bits and for an output buffer rate of 3.5 bits the peak buffer fill is 19,415 bits.

Figures 3.4-8 and 3.4-9 are based on scene 23 which contains sensor data anomalies in spectral band 2. This scene produces an average compressed data rate of 3.28 bits. Figure 3.4-8 dramatically illustrates the effects produced by the defective sensor in band 2 as peaks in the buffer fill occurring every sixth scan line. If the sensor defect were not present the average compressed rate would be around 2.85 bits, well below the buffer output rate. Due to this defect the peak compressed data rate exceeds 4 bits on many of the affected scan lines. The peak buffer fill is 1213 bits. Figure 3.4-9 illustrates the use of a buffer with output rate (3 bits) below the average compressed data rate. The buffer fill continues until it reaches 32,003 bits at the end of the scene. The effect of the sensor anomaly is also evident in Figure 3.4-9 as peaks and valleys superimposed on the buffer statistics.

The Rice algorithm uses a fixed block size of 16 pixels (64 symbols) along a scan line for all scenes processed. This block size was determined early in the investigation by processing segments of three subscenes (numbers 3 , 10, and 20) with varying block sizes. The results are given in Figure 3.4-10. While the performance of the Rice algorithm is dependent on the block size used, and ideally this parameter should be adaptive, compression is only weakly dependent on block size over the range of twelve to twenty pixels per block. As the block size decreases below this range the contribution of the overhead bits becomes a significant percentage of the overall bit rate. If the block length is too large, the symbol statistics can vary significantly over the block and degrade the compression achieved. The block size of 16 pixels was selected as a compromise between these two conflicting requirements and it is felt that any degradation that results fom this choice is not severe.

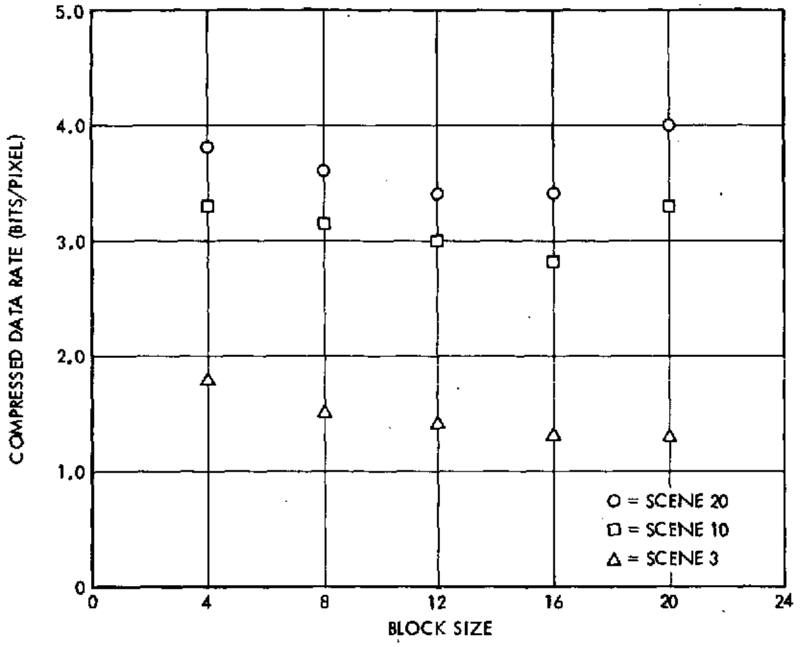


Figure 3.4-10. Performance of Rice Algorithm with Variation of Block Size

Table 3.4 shows the precentage distribution of Rice modes for several scene processed, reflecting the data activity within that scene. For scenes having low data entropy, mode C$\overline{\text{FS}}$ predominates; for scenes having moderate data activity, mode FS predominates; and for scenes of high data activity mode CFS predominates. The only split-pixel mode which appeared in the scenes processed is the (6,1) mode. This is to be expected since the other split-pixel modes appear only when the symbol entropy exceeds five bits.

Table 3.4. Percentage Distribution of Rice and Split-Pixel Modes for Selected Scenes with Varying Data Activity

SCENE No.	FS	CFS	C$\overline{\text{FS}}$	SP(6,1)
1	35.6	3.2	61.2	—
3	37.1	—	62.9	—
13	36.5	63.5		—
15	19.8	48.0	32.2	20.7
24	16.0	84.0	—	—
28	7.3	92.5	.1	—
31	10.4	89.6	—	.2
33	9.1	90.8	.1	8.2

3.5 COMPARISON OF SPACECRAFT AND GROUND PROCESSED DATA

Scene number 28, containing mountains, was run using both MSS tapes before and after ground processing. The former tape contains data generated directly by the spacecraft and permits a comparison of the data statistics and compression algorithm performance with the ground processed tapes which were used throughout the study.

The means of the data differ slightly and are:

	Band 1	Band 2	Band 3	Band 4	Average
Spacecraft	32.86	31.00	39.74	20.27	30.97
Ground	34.96	33.50	38.81	19.28	31.64

This difference is due to the decompression performed on the ground. Figure 3.5-1 contrasts the cross spectral-spatial correlations of the two forms of data and shows a higher correlation for the spacecraft data. Figures 3.5-2 and 3.5-3 show corresponding plots of the joint probability function of first differences from bands 3 and 4, illustrating a somewhat closer clustering of differences about the origin.

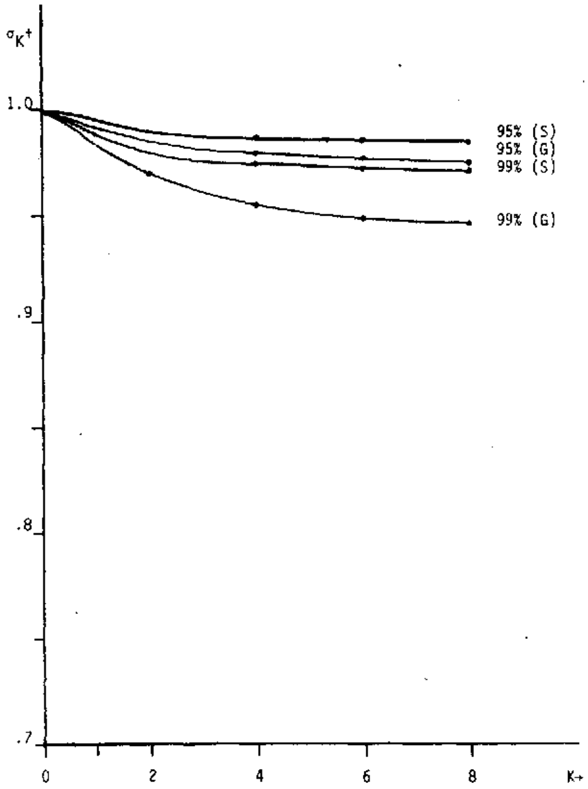


Figure 3.5-1. Spectral-Spacial Correlation of Scene 28 Based on Spacecraft (S) and Ground-Processed (G) Data

	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-16																																	
-15																																	
-14																																	
-13																																	
-12																																	
-11																																	
-10																																	
-9																1	2	2	2	1													
-8																2	2	2	1														
-7															1	4	7	7	5	3	1												
-6															2	4	6	5	4	2													
-5														1	5	14	24	19	12	6	3												
-4														1	3	8	11	8	5	3	2												
-3													2	5	15	39	58	46	22	9	4	2											
-2														1	3	6	11	8	5	2	1												
-1												2	4	11	33	72	100	66	32	11	4	2											
0														2	4	7	9	6	3	1													
1												1	3	10	24	48	57	35	15	5	2												
2													1	3	5	10	11	5	3														
3													2	6	12	20	21	14	5	2													
4													1	2	4	6	6	4	2														
5														2	4	6	7	4	2														
6														1	2	2	3	2															
7															2	2	2	1															
8																	1																
9																																	
10																																	
11																																	
12																																	
13																																	
14																																	
15																																	
16																																	

Figure 3.5-2. Joint Probability Density of Band 3 and Band 4, Scene 28 (Spacecraft Data)

	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-16																																	
-15																																	
-14																																	
-13																																	
-12																																	
-11																																	
-10																	1																
-9																1	1	2															
-8																1	2	2	2														
-7															1	3	3	3	2														
-6															2	5	8	7	4	2	1												
-5															3	4	7	5	4	2													
-4														2	5	12	21	17	9	4	2												
-3														2	4	7	12	9	5	2	1												
-2												1	2	5	14	32	48	40	19	8	3	2											
-1															2	7	16	9	5	1	1												
0												2	4	11	27	60	100	62	29	11	5	2	1										
1														1	4	8	15	7	3														
2												1	3	9	21	39	46	31	13	5	1												
3													2	3	5	9	12	6	3	1													
4													2	5	12	16	18	13	6	2													
5													1	3	4	7	6	4	2														
6													1	2	4	6	7	4	2														
7														1	2	2	3	2															
8															2	3	2	1															
9																2	1																
10																																	
11																																	
12																																	
13																																	
14																																	
15																																	
16																																	

Figure 3.5-3. Joint Probability Density of Band 3 and Band 4, Scene 28 (Ground-Processed Data)

The most interesting comparison is based on the compressed bit rate achieved by each algorithm on the two forms of MSS data:

	Spacecraft Data	Ground Processed Data
SHELL	3.512	3.654
SSDI	3.566	3.698
SSDIA	3.448	3.533
SSDIAM	2.927	2.907
SSDI/Adaptive Huff.	3.33	3.45
SSDI/Rice	3.32	3.44

The compression achieved for spacecraft data was higher for all techniques except for the SSDIAM. While the discrepancy for SSDIAM processing is not fully understood, the improvement achieved with ground processed tapes may result from the compensating interaction between the SSDIAM intensity mapping and the intensity mapping performed during ground processing. The buffering statistics are correlated with the difference in compressed bit rates for the two forms of data, with peak buffer fill tabulated below.

Output Buffer Rate	Spacecraft Data	Ground Processed Data
3.0	36,820	49,875
3.5	4,188	5,734

3.6 RECONSTRUCTED IMAGERY

Four full scenes were compressed and reconstructed using a variety of strictly information preserving algorithms and the Principal Full Scene (PFS), number 31, was compressed and reconstructed to evaluate the effects of channel errors and essentially infomration preserving distortion on the reconstructed imagery. Photographs P1 through P17, contained in Appendix A, were generated from the reconstructed data.

The photographic imagery was reconstructed from tapes formatted as shown in Figure 2.4-5. Several precautions were taken to insure uniformity of film and print processing. All four spectral bands of a given full scene were placed on the same film negative and developed simultaneously. All prints were developed at the same time as were the enlargements of individual spectral bands to prevent gross variations in contrast and to insure a uniform enlarging size.

Photographs P1 through P4 give the four spectral bands of the four full scenes compressed and reconstructed using various combinations of strictly information preserving algorithms evaluated in this investigation. Image P1 contains the scene, number 31, containing a variety of object classes including agriculture, mountains, barren soil, and a river. An airport is located in the top section of the images, on the left side of the agricultural area. This scene was selected as the principal full scene (PFS) since it does contain such a variety of object classes and areas of varying degrees of data activity.

Image P2 is based on scene number 32 containing part of Lake St. John, the Saguenay river, forest, and the city of Alona (Quebec), Canada. Image P3, from scene number 33, predominantly contains coastal vegetated mountains of Southern California interspersed with small lakes. Image P4, from scene number 34, contains the Mojave desert and scattered irrigated agricultural areas.

Image P1 was compressed and reconstructed using the SSDI/Rice algorithm, image P2 used the SSDIA/Rice, image P3 used the SSDIA/Huffman, and image P4 used the SSDI/Huffman. This selection of compression algorithms effectively illustrates the capability of all four combinations of techniques to provide the strictly information preserving compression and reconstruction of MSS data.

Photographs P5 through P8 contain enlargements of the individual spectral bands of the principal full scene. These enlargements are based on the original MSS tape data corresponding to the full scene processed by the various algorithms and provide a basis for comparison of the original imagery with the processed and reconstructed imagery contained in photographs P1 and P9 through P17.

Photographs P9 through P16 contain the individual spectral bands of the principal full scene after compression and reconstruction by the essentially information preserving SSDIAM/Rice algorithm. For photographs P9 through P12, a mapping of $|m| = 1$ was used and each reconstructed intensity level is either the same as the original level or deviates from the original by +1 or-1 intensity levels. A mapping of $|m| = 3$ was used for photographs P13 through P16 allowing each reconstructed intensity level to deviate as much as three intensity levels from the original sample level.

The strictly information preserving compression of the scene produced an average bit rate of 3.59 bits/sample and no distortion. With $|m|= 1$ the average compressed bit rate was 2.71 bits/sample and a mean square distortion of 0.0112 percent. With $|m|=3$, the average compressed bit rate was 1.91 bits/sample and a mean square distortion of 0.103 percent. Figures 3.6-1 and 3.6-2 give error plots showing the magnitude of the deviation between the original and reconstructed intensity levels for a portion of band 3 of the principal full scene. Figure 3.6-1 is based on the mapping $|m|=1$ and Figure 3.6-2 is based on the mapping $|m|=3$

The photographs show that no noticeable visual degradation results from the use of mapping $|m|=1$ since this level is comparable to the system noise including that introduced by the ground processing algorithm. A mapping of $|m| = 3$ does introduce visual degradation especially in areas of uniform intensity such as the valley floor to the left of the agricultural area. This contouring is most noticeable in spectral band 4 (infrared). Fortunately, the algorithm tends to reproduce areas of medium and high data activity well without severely degrading edges or introducing slope overload and overshoot as often occurs in delta modulation algorithms.

Photograph P17 illustrates the effects of channel errors on the performance of the strictly information preserving SSDI/Rice algorithm. Photograph P17 is based on a bit error rate of 10^{-5}. This value was used since it is felt that channels used for the transmission of ERTS data will have an error rate better than 10^{-5}. Errors were simulated on the computer have an error rate better than 10^{-5}. Errors were simulated on the computer by changing bit values in the compressed data bit stream at the appropriate

```
1011000110100111000011100000010100011010000011100010010100000100000000000110110000100110010010001 0001
1010110111001000101111111110010010101000010111100111110101100110001111011011011101101111010111101 0100
1100001000001001100111111111010000100101110110000000111011111111001011111111010111110000000001010001101
1010000011101010111001011001101111010111100001000000010000000001010111100010110001100001111101011 00110
1111111111111111110011010000000001011100111011011111111001111111110111111111111011101111111001100000010110
0000110000000011000101011111010011110000000001000000011101111011111000111110010111111111111101100111000
1111110100001001110010010010001000011001101001111111111100101110111101111101010001011011000000001111011
1011111111101000100100110100000110111111100110000010001010110010100011111111111010110010100100010011
1100000001011110111111001110011010111110000000000001101101000000110000001011010110000111011011100000
0111101111101100100000100110101000111010111100000000100001110010100100110001001001111111101100111111 0
0010111101110001011011000011101110100011000100110010111010011111011110000110001111111100001011111100011
1100101101111110100110110001110001101011111111100000010101000101011111111101010011111111111001111111001
0100011010011010011010001111010000101000000101101011010110000100110111101010000000000000101100000101
0111001110011001101001111111001000100111000100010011101001011110001001111010111111111111000101111011111
1001100111000111101111111100110011111010111110000111111001100111001111111100001000000101101010000111
1100001000110010100111011111000010010011111110011011100110100010000110011010111111111111110011100000 0
0100111001110010100111010110100011110001111010001000000010100000111101001101111111100000010011001001 0010
1111111110011101011000010110101101111111101101111011011011001101011000111111111011111111111110011011111
1111101101001110011100100001001000001111110100000011001001001000001100100010010000000010000111010 1001
1001101000100110101000011011000110110111011110011011101110001111111111001100110001111100100111001 0011
0110011111011011100100011001001111011011111111111110101000011011000101011111111111111111101001111110
1100101111101010010101001111001111100110111000011111111110111011100000110100001111000001110010100 10101
1111111110011001110111111111111011101001110011111000011000100000000111011111010111100101010000 01000011
0100011100001010101010000110011111110000000111111110011111100101110110001100100000001011001111101101011
0110001000011010010001110010000100101111110100101011111111110000111010100000000010001001010100001001011
1011111111111100010000100011000001110000111111111100110011001001010010111010101100101011100101001 0011
1110010011110010011000000000000000000000000000001011011011111111111111111111111110101001100100111101111 1
1010111000110001010101100101110000011111111111110010000000010010101001110101111011111100101111100000 01
0001011011111000011100111011100010101110100111110100101110010000000000111010100101000001110000100110 00
1101110000101001011011111110101110101001011111100011100111011111111101100010011011011110101101111001 11
1001101111000100010101101011010001000010100001011010000101110101000110100011111110010111011010000000 00
0101111011111011010101110100000101010011011111010000000001111111110000011010011110101011001101 1001100
0011001001001001100001000110000011001010000110111010000000111100000111000110100000000000001111011101
1011000110011000001010111110011101001111010101111101111110111000000111100010011010011100010000000000
0001101111001101100010011101101010010001111101010010010001111011111111100111100110110101001100110 0
1110001010100110011000010010110001000010111100000000010101001011000000111001000001101011100101001111 0
0101111010001101011111111111000101011000000101111110101000111100111100100111100000111101100111111001 0
0001111000100100110111111111101010011110110111110100110101101101011110010000111101011100010000011000
1001101110001001001101000100101001000111111100100011000111111110000010101000101011001101010011111111 1
1111100100100011100111111000011111001011010011111000000110101010101100010101000100000000101110110101
1010100110111111111111111111111011101011111000100000000110100011111110100000000000000000101110100000001
0101110111000110010000000000000000000010100110110101000010001111100111011111101100011011111110001111010
```

Figure 3.6-1. Error in Levels Between Original and Reconstructed Samples in Band 3 of Scene 31 (|m|=|)

```
3011000130302311020213300202010300031212002013120010212102202302200022021103302203201300320320032221
3232112333223002101133333320122303012222103313201331323033201122233132110312313031011132123131230300
1122001000001201122131311130120201003011121122220223330331131122323311311012111112000000001010201303
12322222133210321132210330011033333230111120001220220120002000301013332201211202130200113310101120332
3131111111133133111021121020220221211100111031013111133221111113013111333131213121111111221102022232132
0000330000000011222323213331232033112000000212020202331231112131332201113023013331113311321322111000
3333112322023203312030230212203020033001101021331113111320121332331301111030322012312110000220311121 1
12111111133323202122120130122020110133313100332202230201032332232120213131111331010130032100122010233
110222222121113013113100113021323233111000220020000221101101020220330020010130301320021332112111122020
011110111121300320220300110103000111010131320020222320021132212122122132201001201333333101300333333 0
0010111112311022101101120001330111212023320230213003011121201331011312020110001133333000210311132003 3
1122321301133301223323122211302211210133311113020200321212223012333313332301201111133331021113333021
0300031230211010211232221333030002103002020103121213212110020100112111101212022222222000323120222101
2311003132211023101001113110210021023330001220102111030212113322232233332323131111313300212331231133
120132231320211130311311120112231131212131110222331113122312031122111221111113222212222221231010222113
11000010223102123003330131113002210010031331110011013322330120212222112013030333211113111110231102202 2
01201130233100121203310121321000333100031310102210002021012022111121223303311333002203001322122100 12
3111311131203330321322201013232110131131112312111123101101122332323300211113313031311131113122132113 31
13111011210011320111221000032032000233311301222220330010030010022211001200300102200221020213303210 21
120330322230211032100221123322231033011301133201101312313022313331133201320130001111302102111023023 1
01322133130112133001202332232231132112313333331313301212022332310221010113331313311313131212011131 30
3102230111103212021012233130011131023101330220331311311011121110202211010000113300222131221210010303
1133113112211023130113111331312113012213300333102223302212020020311011111010331100101012220210000 11
21020333022012123012320023120113313000202233313310011111221031121122233223222000212112231131211210 11
013000320021103221222113201022212232113111232230101111111302003132321200222001002120103010220320103 3
12131311331111200102021022312220231320023133333111221122312010232322123321232132221210113201232210031
1132210011130230213220202002020222222200022202003233213011131131331331113333301210033001001133031113
1010313020112003232123302301110202013311311131111021222002210210121003112301333011331003011111000000 1
20230310111120201130213103312203230113010031113012212311001222222222311010100321022001112000100110 20
1323112222303021213013111131232111121230232133313202111001110311111130110221221103323333212112333223 11
300332111100212221212110101123000120021030220323301202010111010302013232203331112012111031212220002 0
0123111233333013030321132120222323012211211311230220020003311111120000112302111121212332233233221102
2211221021221221302001200110020211201010002312111012002203313202021110001121222220222000223133211321
301100211223322220032121113102333030213112321231113211331101132202021111020122330322333000102200022 0
00031031112013031202102113011212120100013111301210010210012201133031111110011112213011210102110213 00
1132023012302112013202030030110001022210113100000000210121021011000003332230022231230331221212231310
2123131012201323033311131132223210330000001213313123212001333023333021223111022203331231023333310032
2021333020100300330313133331121012011130130133130122312323323123301131001222211312101132003000031002
1221101112021001221101200122103201002131333100320211222133311120222123212001012330011230100111111111
1131332010212223332211111322201113120301103001131102220013010301210130021210120012020000323310110103
1230320332233111133333111313133312111123013111002102022000112122011113303002202202002020030111010000 2001
01013303110003300300000000000000000002101023301101232222100013130033303333101100213213111120221311012
```

Figure 3.6-2. Error in Levels Between Original and Reconstructed Samples in Band 3 of Scene 31 (|m|=3)

error rate. Such an error affects the reconstruction algorithm and produces anomalous reconstructed data values until the next memory update point where the algorithm again synchronizes with the compressed data and generates correct data until the next error occurs. The current simulation only provides such a memory update at the beginning of each scan line but the algorithm can be easily modified to produce more frequent updates.

3.7 IMPLEMENTATION OF THE SSDI/RICE ALGORITHM

An investigation of the implementation of the SSDI/Rice algorithm was conducted concurrently with the simulation activities to determine the logical data flow for such a system and estimates of hardware complexity. Although the baseline system developed could not be optimized within the scope of the present study, the investigation performed serves to provide a basis for further study and refinement and permits tradeoffs pertinent to decisions involving the use of such a data compression system on board future earth observation satellites.

The SSDI/Rice Data Compression Unit (DCU) functional block diagram is given in Figure 3.7-1. A flow chart of the DCU information transfer and processing is illustrated in Figures 3.7-2 and 3.7-3. Figure 3.7-4 through Figure 3.7-7 depict the basic hardware components necessary to implement the SSDI and RICE algorithms. The data flow begins in Figure 3.7-4 and continues to Figure 3.7-7 with the functional blocks listed identifying the relationship to the block diagram in Figure 3.7-1. The notation used in labeling the components is listed in Figure 3.7-4. The baseline implementation utilizes SSI and MSI Low-Power Schottky TTL (SN54LSXXX) combined with bipolar 1024 x 1 RAM's. Custom LSI (TRW Emitter Follower Logic) was investigated for repeated functions such as the ALU/Latch. The estimated impact was principally in a reduction of total parts and power of 10% and 0.8W respectively as compared to the baseline. Since the development cost of the LSI chip is sizable and the net effect on the DCU is minor, LSI was not included in the baseline design.

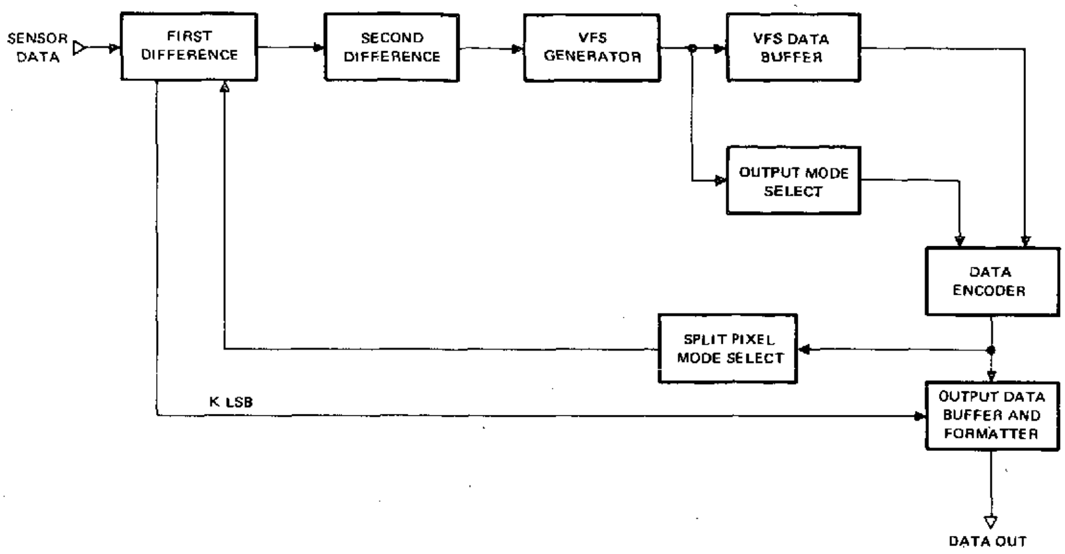


Figure 3.7-1. Overall Block Diagram of SSDI/Rice Data Compressor Unit

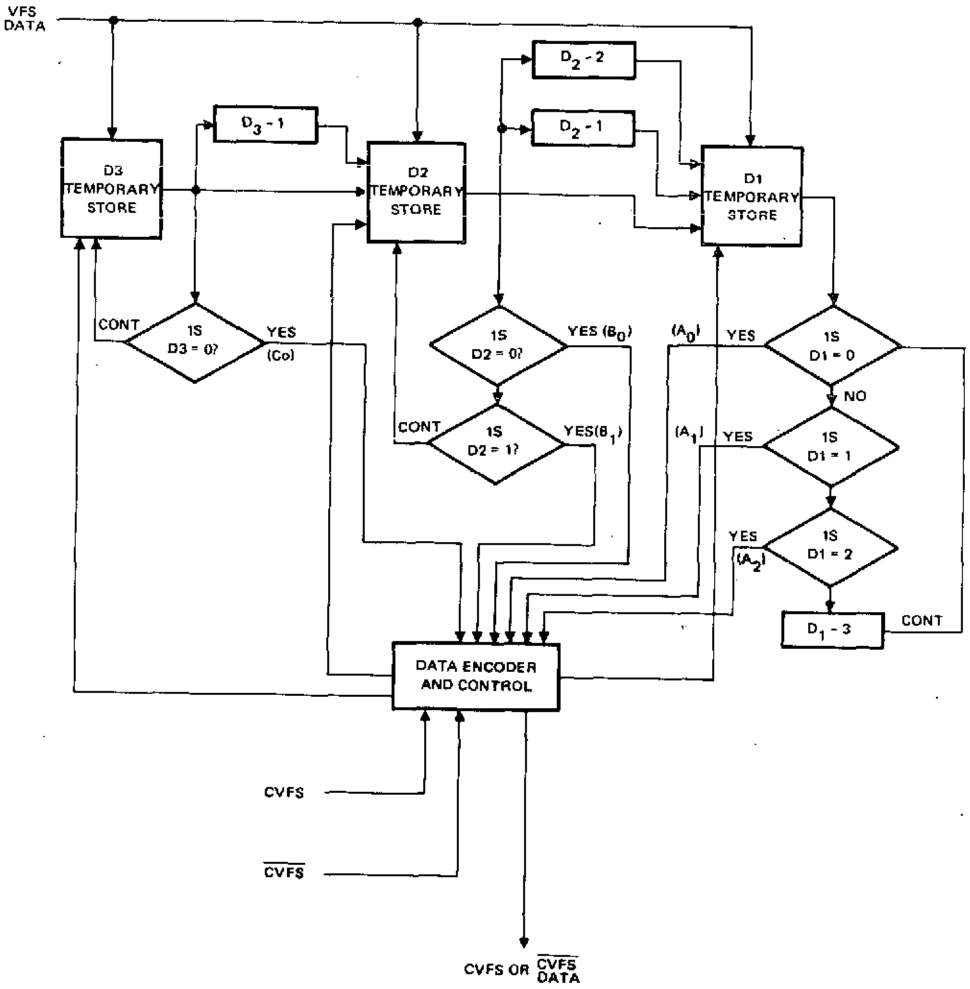


Figure 3.7-2. Flow of Data in Code VFS Operation

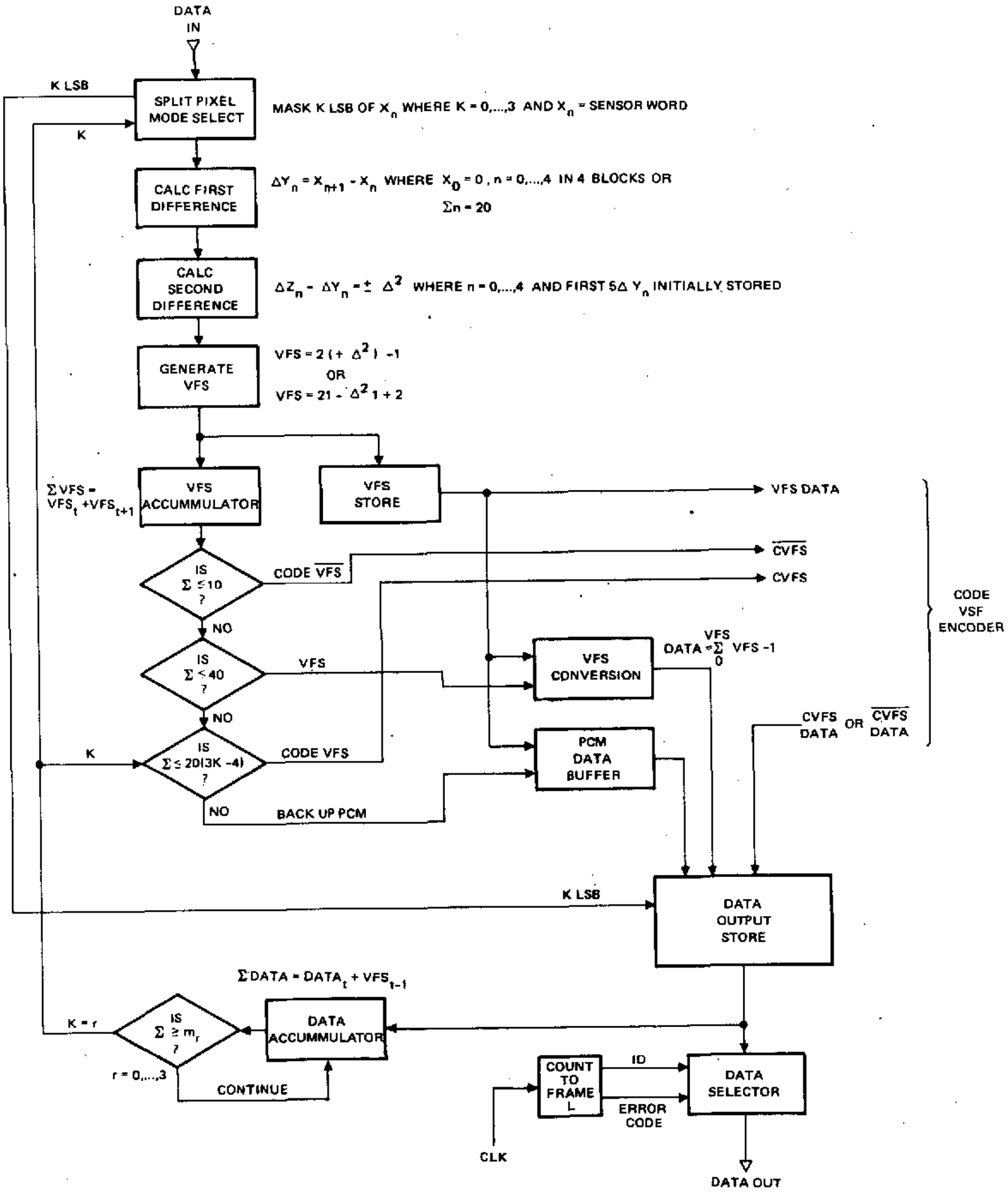


Figure 3.7-3. Flow of Data in DCU

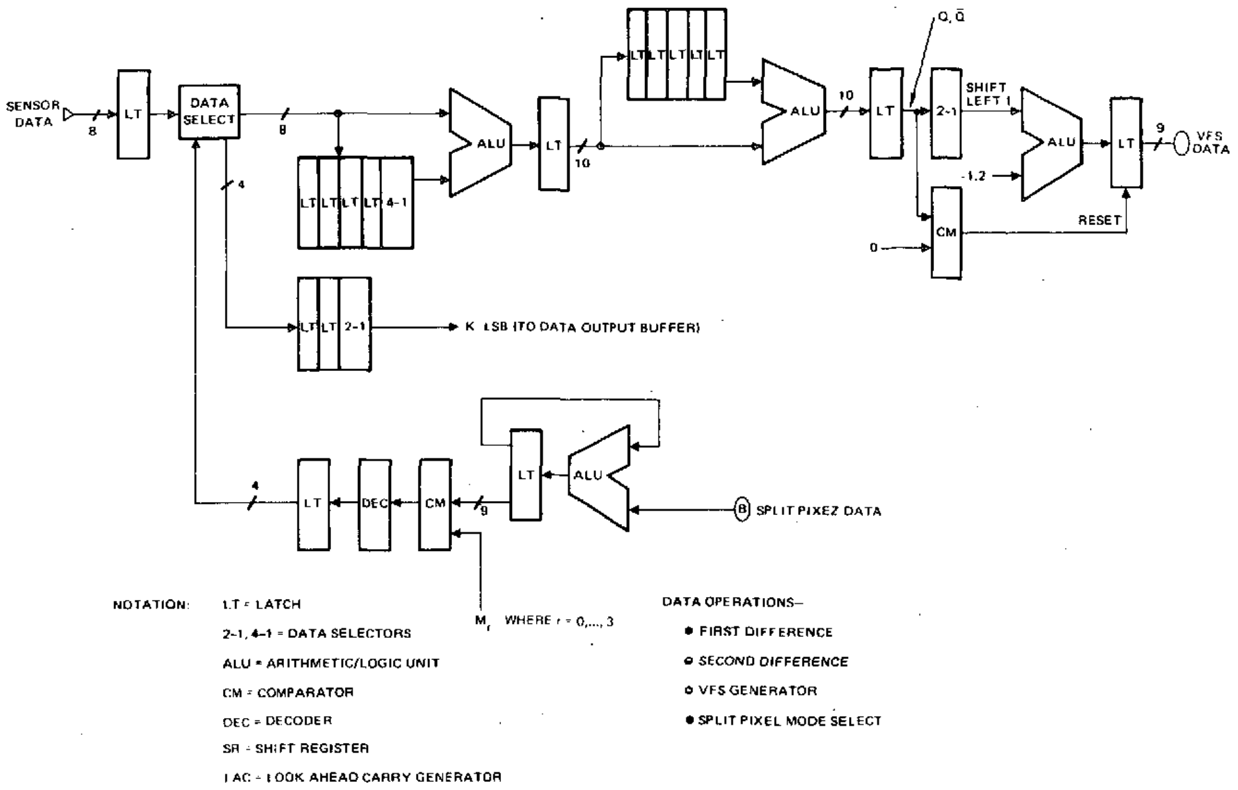


Figure 3.7-4. Data Operations — First Difference, Second Difference, VFS Generator, Split Pixel Mode Select

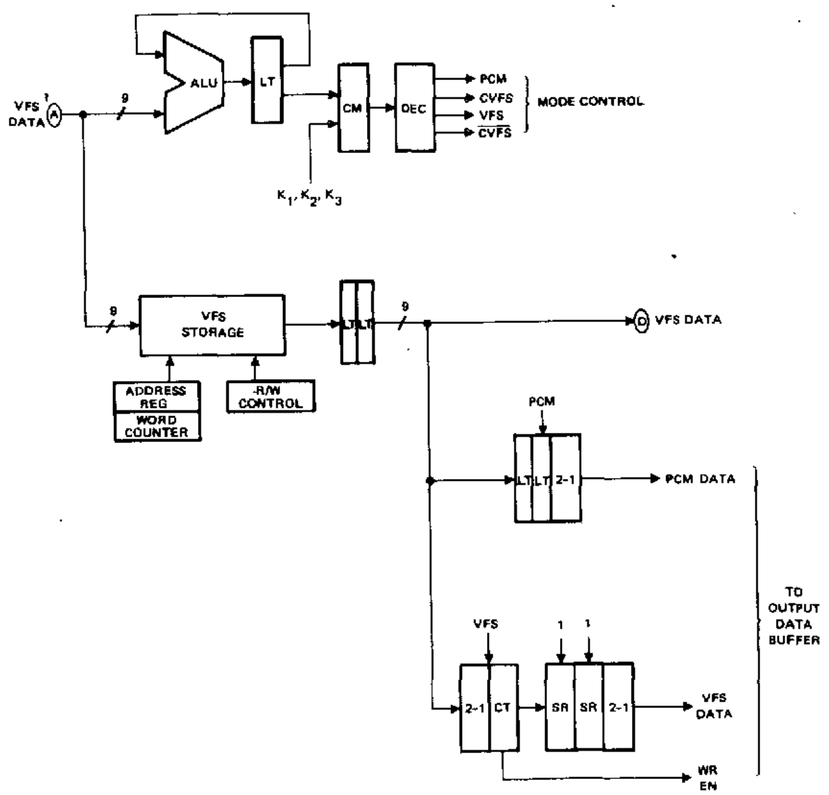


Figure 3.7-5. Data Operations — VFS Data Buffer, Output Mode Selection

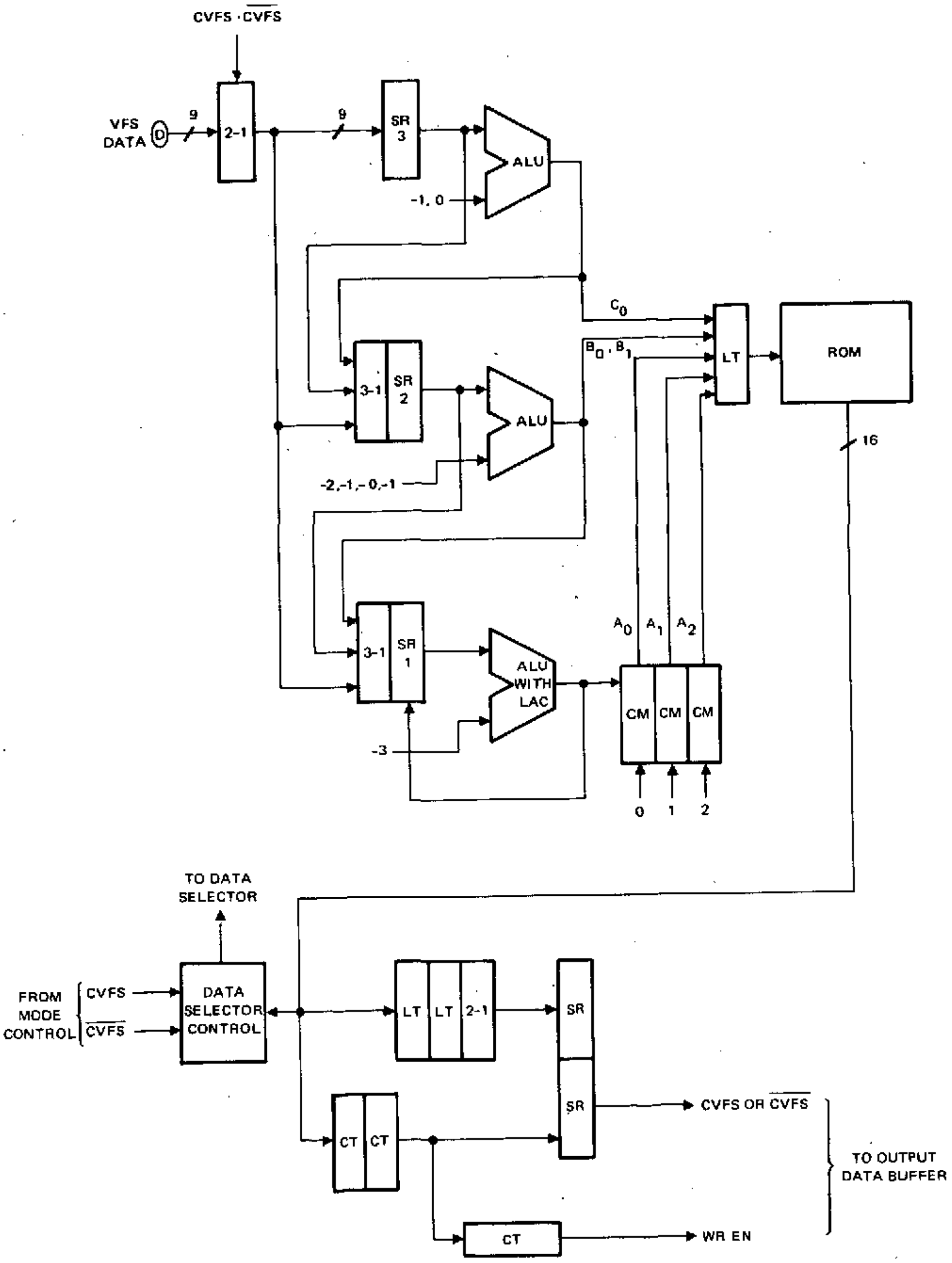
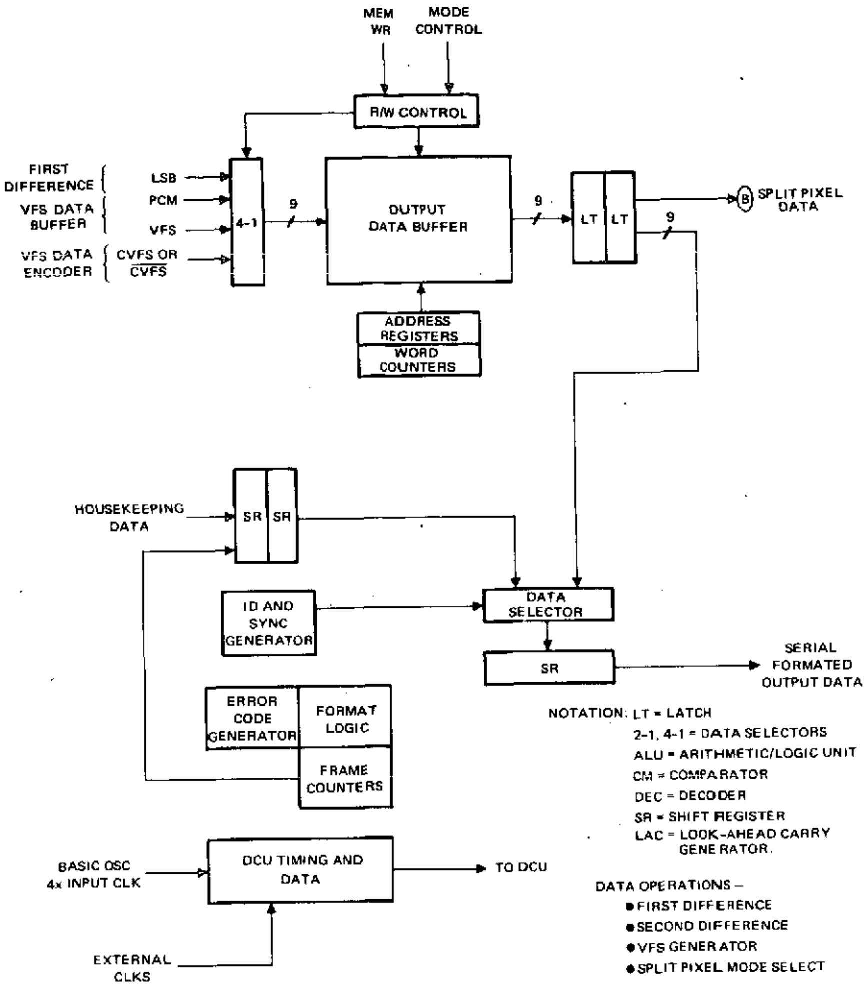


Figure 3.7-6. VFS Data Encoder Diagram



NOTATION: LT = LATCH
2-1, 4-1 = DATA SELECTORS
ALU = ARITHMETIC/LOGIC UNIT
CM = COMPARATOR
DEC = DECODER
SR = SHIFT REGISTER
LAC = LOOK-AHEAD CARRY GENERATOR.

DATA OPERATIONS –
- FIRST DIFFERENCE
- SECOND DIFFERENCE
- VFS GENERATOR
- SPLIT PIXEL MODE SELECT

Figure 3.7-7. Data Operations – Output Data Buffer, Data Formatting

The baseline DCU without a power converter is summarized below:

- Data clock rate 5 MHZ (equivalent data rate 35M B/S)
- DCU Parts Total: 273 IC, 60 Discretes or 323 total
- Power: 26.7 watts
- Weight: 3.0 pounds
- Volume: 192 inches^3 (6" x 8" x 4").

The maximum clock rate capability is 10 MHz for the given design. The rate can be further increased to 15 MHz maximum if selected components are substituted with standard Schottky TTL MSI and speed enhancement IC's used for Look Ahead Carry Generators for the Arithmetic/Logic Units (ALU's). There is then a penalty in additional parts and power dissipation with a corresponding slight increase in packaging.

The unit characteristics for the baseline configuration is tabulated in Table 3.4, for three types of units; I for 10 MHz, II for 15 MHz, and III for 25 MHz. Type III was sized using Schottky TTL with Emitter - coupled Logic (ECL) at critical points. The resulting characteristics, especially power, are very large and are shown for comparative purposes. An optional power converter is also listed in the same terms as the DCU and would be directly additive to the associated DCU type. Example: Power (Type I) = 26.7 + 9 = 35.7 watts total.

While Table 3.5 shows three technologies providing increasing input data rate capabilities of up to 175 Megabits per second, a severe penalty in power, weight, and volume is imposed by a type III implementation. If the system must run at data rates higher than 105 Megabits per second a different form of the SSDI/Rice should be developed so that certain system blocks process separate blocks of input data in a parallel fashion, later reassembling the data into the appropriate serial bit stream. The output data buffer size is a function of the buffer output bit rate and should ideally adapt its parameters to the time-varying compression statistics. A number of developing technologies, such as bubble and CCD memories could permit the use of buffers having millions of bits of storage and occupying a relatively small volume.

Table 3.5. DCU Unit Parameters

TYPE	INPUT CLOCK RATE (MHZ)	INPUT DATA RATE (MBPS)	PARTS			POWER (WATTS)	NUMBER OF SLICES	WEIGHT (LBS)	VOLUME (IN^3)
			IC	DISC	TOTAL				
I	5	35	263	60	323	26.7	2	3.0	192
I	10	70	263	60	323	26.7	2	3.0	192
II	15	105	290	60	350	30.7	2-1/2	3.7	192
III	25	175	333	160	493	108.0	3	4.5	384

POWER CONVERTERS

TYPE	PARTS		TOTAL	POWER (WATTS)	NO. OF SLICES	WEIGHT (LBS)	VOLUME (IN^3)
	IC	DISC					
I	10	240	250	9.0	1	1.5	96
II	10	240	250	10.0	1	1.5	96
III	20	250	270	36.0	1	3.0	128

Notes: 1. Slice form factor 6" x 8" x 2"

2. Power converter is optional, hence, add as necessary. Assumed efficiency if 75%

4. SUMMARY AND CONCLUSIONS OF THE INVESTIGATION

This section summarizes and discusses the major results of this investigation and the relevance of the study to the ERTS program. The intent of this section is to provide information which can assist in the planning of future ERTS missions relative to the use of data compression either on board the satellite or for ground-based applications.

4.1 DISCUSSION OF RESULTS

The information obtained by the processing of thirty-four ERTS MSS scenes permits the drawing of certain conclusions regarding the relative performan of the several data compression algorithms evaluated. Since the results obtained are based on the necessarily finite amount of ERTS scenes used, the conclusions reached are strictly valid only for this data but are representative of the level of performance to be expected for similar data. For example, if a given object class subscene yielded a compressed bit rate of 2.5 bits/sample, other subscenes containing this class should produce similar results. The variance of data statistics should also be minimal for similar object classes, assuming that the data does not contain sensor or processing anomalies such as that produced by the defective sensor in spectral band 2.

4.1.1 Data Statistics and Compression Performance

The MSS data used was based on ground processed tapes and one unprocessed spacecraft tape. The ground processing includes a decompression algorithm which performs a mapping of intensity levels in bands one through three. The effects of the mapping can be seen from a comparison of the probability density of the intensity levels taken from a ground processed tape, Figure 3.3-3, and the corresponding distribution of intensities from the same segment of data before ground processing, Figure 3.3-2. The processed tape shows a redistribution of intensities with a corresponding omission of certain intensity levels present in the spacecraft data. The effect of ground processing on the performance of the compression algorithms will be discussed later in this section.

Some of the scenes processed reveal the presence of anomalous sensor data in spectral band 2. As shown in a segment of data values in band 2 of scene 8, Figure 3.3-4, the anomalous data occurs every sixth scan line when present. Such defective data is uncorrelated with the data in the adjacent scan lines and with the equivalent scan lines in the other spectral bands, producing a degradation in the performance of the data compression algorithms which require a high degree of spatial and spectral correlation in order to obtain good compression.

For data tested, not containing this anomalous data, the measured data statistics were fairly accurate indicators of data compression performance, except for the variances of data intensities. This variance measures the global variation in intensity over the scene,whereas the performance of the compression algorithms is dependent on local variations. Therefore, a scene for which the intensity levels are very uniform over local areas while the mean intensity changes dramatically over large regions can exhibit both a high variance and a low compressed bit rate. Conversely, a scene which exhibits a low global data variance in all spectral bands produces a low compressed bit rate. The joint probability distribution function of first difference values is a better indicator of the degree of compression possible for a scene, with compression increasing as the clustering of joint differences tightens around the origin. The cross spectral-spatial correlation is a valid indicator of the average spectral correlation of the scene over local regions. The less rapid the fall-off and the closer the spacing of the 90%, 95%, and 99% correlation curves, the higher the compression.

Table 4.1 summarizes the average compression obtained with each algorithm for the uniform object classes evaluated. For multiple scenes containing the same object class, the range of compression results falls within ten percent of the average value, except for the object class labeled forests where the compression deviates up to thirteen percent from the median value. Such differences reflect variations of location, sun angle, and time of year among the scenes processed. The compressed bit rates generally reflect the level of activity in the scene, progressing from the low data activity seen over large bodies of water to the high activity occurring from field-to-field in agricultural areas.

Table 4.1. Average Compressed Bit Rates for Uniform Object Classes (Based on Scenes Processed)

			GLOBAL HUFFMAN			
OBJECT CLASS	AD. HUFF.	RICE	SHELL	SSDI	SSDIA	SSDIAM
CLOUDS	1.42	1.31	1.697	1.508	1.491	1.427
WATER	1.51	1.40	1.956	1.921	1.635	1.550
SNOW	1.78	1.79	2.014	2.135	1.890	1.453
PLAINS	2.55	2.67	2.827	2.790	3.251	2.724
DESERT	2.74	2.84	3.081	2.964	2.983	2.484
MOUNTAINS	3.23	3.24	3.403	3.500	3.362	2.798
CITY	3.06	3.09	3.379	3.373	3.343	2.731
FOREST	3.10	3.11	3.339	3.370	3.368	2.891
GRASSLAND	3.32	3.35	3.581	3.536	3.487	2.981
AGRICULTURE	3.41	3.42	3.640	3.566	3.593	3.042

The entries in Table 4.1 can be used as a guide for determining the expected level of compression for another scene containing the same object class. The compression of a full scene of data can be roughly estimated by weighing the averaged compressed bit rate of each object class contained in the scene by the percentage occurrence of the class over the scene. The resulting rate should be fairly close, especially if the adaptive Huffman or Rice algorithms are used. The estimate could be too low for the global Huffman algorithm if the bit rates of the object classes in the scene substantially differ, as in the case of a large lake surrounded by agriculture or forest, since the code developed for the entire scene may be a poor match to local areas. The estimate should also be tempered by the presence of haze, smog, or clouds. Haze and smog, improve compression by lessening the apparent data activity as seen by the satellite whereas small broken clouds can produce large intensity jumps at their periphery, decreasing compression. Large cloud cover, however, can be considered as a separate uniform object class when performing such estimates.

Based on the scenes processed, the average bit rates for the various compression techniques are as follows

	Global Huffman				Adaptive Huffman	Rice
	SHELL	SSDI	SSDIA	SSDIAM		
Average bit rate	2.99	2.98	2.92	2.50	2.67	2.70

Based on the actual compressed tapes generated for the 25 x 25 n mi full scenes, a compression ratio of 2:1 produces a compressed bit stream that fills 7.1 percent of a standard reel of magnetic tape. This result and the average bit rates achieved for the scenes processed indicate that almost all 100 x 100 n.mi scenes could be compressed to occupy a single reel of tape as opposed to the four reels now required. This result can be of great economic benefit to NASA for storage, transmittal, and achieving of ERTS imagery data. With an essentially information preserving algorithm such as the SSDIAM an even greater compaction of the data would result.

The above measures of data compression performance and compressed bit rate are based on the use of ground processed tapes. One tape of spacecraft data, before ground processing, was received and evaluated. The results of processing a given subscene of this spacecraft data and the same subscene after ground processing indicated a lower compressed bit rate with the spacecraft data for all the strictly information preserving data compression techniques used. This decrease in bit rate is probably due to the higher correlation of the spacecraft data which does not contain the deleterious effects of the ground decompression mapping of intensity levels.

It is not possible to decisively conclude that all spacecraft data will produce a lower compressed bit rate than the corresponding ground processed data on the basis of the single spacecraft tape processed. However, since the bit rates differed by only a few percent, it is reasonable to conclude that the results achieved by the processing of ground processed tapes during the investigation are applicable to spacecraft data with only minor changes in the bit rates achieved with each source of data.

4.1.2 Effects of Distortion and Channel Errors

The SSDIAM/Rice algorithm was used on the principal full scene to evaluate the effects of essentially information preserving photographs P9 through P16 of Appendix A. The results indicate that a substantial decrease in bit rate can be achieved by this form of data compression without introducing a severe distortion of the reconstructed image. For the example illustrated, the strictly information preserving compressed bit rate was 33% larger than the SSDIAM bit rate for mapping $|m| = 1$ and about 90% larger than the SSDIAM bit rate for mapping $|m| = 3$. The distortion introduced for mapping $|m| = 1$ cannot be discerned visually when compared with the original imagery and the effect on the data is comparable to that introduced by decompression during ground processing. The distortion produced by the mapping $|m| = 3$ can be seen especially in areas having a relatively uniform intensity where the effect is similar to the contouring often seen in imagery compressed by delta modulation techniques. The effects of distortion are less evident in areas of high data activity and no slope overload or overshoot effects are produced by the SSDIAM algorithm.

An adaptive form of the SSDIAM is suggested based on the results of this investigation. In this study a fixed mapping block size of 8 pixels was used. Ideally, this block size would vary with data activity as discussed in Section 2.1.4. In addition, mapping level m would vary adaptively as a function of both the data activity and mean block intensity level to compensate for the logarithmic response of the eye. Note that these adaptive block techniques only add complexity to the compression algorithm. The reconstruction algorithm is the same whether the compression is strictly or essentially information preserving. While an optimal form of the SSDIAM could produce reconstructed data useful for the majority of investigations involving visual interpretation of ERTS imagery, the effects of the distortion on experiments involving computer processing of the data are unknown at present. Due to the mechanization of the SSDIAM, however, in which the block spectral means remain unchanged by the processing involved, the effects of the mappings may not severely degrade the accuracy of algorithms used for classification of crops. Since the SSDIAM does produce a significant increase in compression, the effects of such distortion on the results of various ERTS investigations should be evaluated.

The compressed data is far more vulnerable to the effects of channel errors due to the removal of the redundancy present in the PCM spacecraft data. Such errors can occur over the transmission link from the satellite to the ground receiver or by dropouts on the magnetic tape storing the data. The effects of bit errors propogate differently depending on the compression algorithm used. For the SSDI/Rice algorithm used in this study, a bit error produces incorrect reconstructed pixel intensities until the next memory update. This is true whether a single bit was changed or a burst of noise affected a number of bits. Since the present simulation updates the memory only at the beginning of each scan line, the number of pixels corrupted depends upon the point in the scan line when the first bit error occurred.

Photographs P17 through P24 shows the effects of channel bit error rates of 10^{-5} and 10^{-6} on the reconstruction of the principal image. Errors occurring at rate 10^{-6} produce only minor effects on the data while errors occurring at rate 10^{-5} can be seen. Since transmission channels and tape recorders used for the ERTS program have less than a 10^{-5} bit error rate, the effects of such errors should not be a limitation on the use of data compression. Moreover, the propagation of such errors can be further limited by the insertion of several memory updates on each scan line. The overhead bits required for four or five memory updates per scan line would only increase the compressed bit rate by a fraction of one percent.

4.1.3 Implementation of the Compression Algorithms

Section 3 contains a mechanization of the SSDI/Rice algorithm and a preliminary sizing of the hardware parameters associated with this system. This combination of the SSDI and Rice techniques appears to be a viable form of data compression for use aboard a spacecraft for several reasons:

- The SSDI is the simplest algorithm to implement and is less affected by sensor anomalies than the SSDI technique.
- The hardware required for the Rice implementation is moderately complex but capable of operation at high bit rates with current technology.
- Channel errors produce distortion in the SSDI/Rice reconstructed data that are constrained to a segment of a single scan line. Use of either the SSDIA or adaptive Huffman techniques require the use of a higher quality channel since errors can propogate over several scan lines.
- The Rice algorithm has the adaptivity required to produce a high degree of compression for data where the source statistics vary considerably over the scene.

If the SSDIA/Rice technique is implemented as an alternative, the complexity required in the processor would increase slightly but an increase in storage would result due to the averages which must be computed using intensities from the previous scan line. In general, an entire scan line of data must be stored in each spectral band to accomplish the averaging. A shift register form of storage would permit the rapid access of the required intensity samples from the previous line. As each new intensity is reconstructed on the current line it is shifted into the register, shifting out the third intensity from the previous line required for the current averaging operation. The averaging of the four samples is easily accomplished by adding the appropriate four intensities and shifting the result two places for the divide operation.

The storage of a full scan line of intensities in each spectral band implies a storage of almost 90 kilobits, a value which may not be excessive in the near future as CCD storage becomes practical. The required storage could be halved by use of a modified SSDIA algorithm in which each two successive reconstructed intensities are averaged and stored as a single number to be used for SSDIA averaging when accessed.

Implementation of the adaptive Huffman algorithm was not studied. The most difficult portion of this algorithm to implement is the hardware required to rapidly convert the probability distribution of symbols measured on one scan line into a Huffman code for use on symbols in the next scan line. This could be done off line by the following sequence of operations:

- Measure symbol probabilities for scan line i
- Generate and store symbols for scan line i+1 and in parallel generate the Huffman code for line i+1 simultaneously
- Encode symbols from scan line i+1 during scan line i+2

This technique requires the storage of two scan lines of symbols at a time but the encoding delay permits each new Huffman code to be generated over the time required to scan a line of new data. A suboptimal alternative would involve the storage of several fixed Huffman codes on the spacecraft and, at the end of each scan line, select that fixed code which best matches the measured symbol statistics for use in encoding the following scan line symbols.

The added complexity of the SSDIM or SSDIAM hardware over that of its strictly information preserving counterpart is minimal. The block averaging required involves simple add operations and, if the number of pixels is a power of two (2^k), k right shifts of the sum. Each intensity sample is then varied up to m levels closer to this block mean before the SSDI or SSDIA operations are executed.

For ground based data compression implemented by computer, a different set of requirements is imposed. The compression should be relatively fast to prevent taking up an excessive amount of computer time beyond that already required for processing the ERTS imagery received. The compression achieved should be sufficient so that the resulting economic benefits offset the additional processing. In addition, reconstruction algorithms should be efficient and capable of being performed by the user of the data in most cases. The SSDI/Huffman algorithm has merit in such applications and a preliminary approach to implementing such a ground based compression and reconstruction technique is outlined in Appendix D.

4.1.4 Relative Algorithm Performance

While no decisive statements can be made concerning the absolute performance of each algorithm because of the finite amount of data processed and the presence of anomalous data varying from scene to scene, certain general conclusions can be drawn. First, the adaptive Huffman and Rice algorithms invariably give a lower bit rate than the global Huffman algorithm based on the same set of symbols. This result is expected since global symbol statistics are rarely optimal for local areas within the scene (unless the scene contains a very uniform object class). The use of an adaptive coding technique is mandatory aboard a spacecraft where all object classes are observed. Due to implementation considerations, the Rice algorithm appears to be a viable since it is capable of operating at high data rates and requires no computation or storage of source symbols beyond those in the block being processed. As shown in Appendix D, the global Huffman code has advantages for the ground compression of individual 100 x 100 nmi frames of ERTS data for which the global statistics are sufficient to obtain a significant compaction of data allowing the savings of several tapes per frame.

Since the performance of the SHELL algorithm is comparable to that of the SSDI and the implementation of the SSDI algorithm is less complex, the SHELL technique has little merit for further consideration. Of the two strictly information preserving algorithms, SSDI and SSDIA, results are mixed. The SSDI provides better compression if the correlation of the data is significantly higher along the scan line than from one scan line to the next. For this reason, the presence of anomalous data, as in spectral band 2, produces a higher bit rate with the SSDIA than occurs with the SSDI. Although the implementation of the SSDIA is not significantly more complex than that of the SSDI, the SSDIA requires additional storage. The propagation of errors in the SSDI algorithm is constrained to a segment of a scan line between successive memory update points but errors in one scan line of data reconstructed by the SSDIA also effect reconstructed values in following scan lines due to the averaging operation performed. This imposes a requirement for a higher quality transmission channel for the SSDIA than needed for the SSDI to give a similar quality of reconstructed data.

The essentially information preserving SSDIM or SSDIAM algorithms yield significantly lower compressed bit rates than the strictly information preserving algorithms. The additional processing required for the mapping operation is negligible and the main consideration is the impact of the distortion on users of ERTS data. From the preliminary results achieved during this investigation, it appears that an improved adaptive form of this algorithm could have potential application in the ERTS program, especially for data compressed and stored on the ground.

Although only one set of tapes of spacecraft data was processed, results indicate that the algorithms give a comparable compression for spacecraft data as for ground processed data and the algorithm performances maintain the same order of ranking. The scene processed produced a somewhat lower compressed bit rate for spacecraft data than for ground processed data. It cannot be conclusively determined from this limited data if this is a general result.

4.2 IMPACT OF DATA COMPRESSION ON THE ERTS PROGRAM

4.2.1 The Requirement for Data Compression in Future ERTS Missions

The multispectral imaging sensors of ERTS-A generate tens of billions of bits daily. In succeeding missions, this figure will likely multiply by several orders of magnitude as higher resolution sensors and more spectral bands are added. Such volumes of data and the implied data rates present severe problems in communication links, in ground data processing, and in ground data storage and archiving. The actual feasibility of including an experiment may therefore be threatened by the large data rate and the accompanying data handling and communication overload. In addition, several digital tapes must be provided each investigator for every scene he requests.

Efficient source coding, i.e., data compression, can yield significant benefits and alleviate such problems by exploiting redundancies in the data to reduce the amount of data which must be transmitted, processed, and stored. The objective of source encoding is the exploitation of the statistical dependence between data samples so that only that information which is essential to the faithful reproduction of the image need be transmitted.

The success of the earth resources survey programs will be measured to a large extent by the satisfaction provided users of the data. That satisfaction depends on how well the data, as formatted and processed, helps the individual accomplish tasks that are significant relative to his goals. Any processing performed on the data should be accomplished without sacrificing the information fidelity required by the user.

Many forms of data compression have been studied, some simple and others quite complex, but not all of these techniques preserve information in the sense that the original data can be reconstructed with arbitrarily small error. A strictly information preserving data compression algorithm provides reconstructed data identical to the digital sensor data entering the compressor. Such a technique preserves the archived data and cannot be criticized by any user as invalidating his data requirements. On the other hand, strictly information preserving techniques are limited in the amount of compression available.

Essentially information preserving compression permits a much higher compression to be obtained but permits a small amount of distortion in the reconstructed data. While an average compression much greater than 2:1 is difficult to achieve with strictly information preserving techniques, essentially information preserving algorithms can yield significantly higher compressions with only a modest degree of distortion. In many cases, if the compression is properly performed, much of the distortion actually arises from elimination of sensor noise rather than by the destruction of useful data. Such a form of essentially information preserving compression yields data which can be used by many of the scientific investigators, and the reconstructed image simply suffers an apparent slight decrease in signal-to-noise ratio which can possibly be improved by postprocessing techniques.

The requirements for data compression are different, depending on whether the initial compression is performed on-board the satellite or on the ground. For on-board processing, size, power, weight, and complexity must be minimized; a more complex reconstruction may be used, however, since reconstruction is usually performed by a ground-based computer. Ground-based compression may be considerably more complex since it would be performed by a large computer which need not necessarily operate in real time. Reconstruction, in this case, usually is performed by the user, and an efficient rapid reconstruction is mandatory to reduce the user's cost. There is a definite need for more than one form of data compression.

4.2.2 Application of this Investigation to ERTS Program

This study concentrated on the low complexity SSDI algorithms combined with source encoding as a technique applicable to either the spacecraft or ground-based data compression of ERTS digitized imagery. The results are quite encouraging regarding the suitability of these techniques for applications in the ERTS program.

The SSDI/Huffman algorithm forms the basis of an efficient technique for ground-based data compression and reconstruction that can be performed with a modest amount of computer processing. Results indicate that the average 100 x 100 n mi scene can be compressed to a single reel of magnetic tape and reconstructed with no loss of information. This reduction in storage from four reels of tape to a single reel yields economic benefits through a reduction in tape costs and in the tape storage facilities required while permitting a simplified archival procedure to be employed.

The essentially information preserving SSDIM or SSDIAM algorithms can yield reconstructed data which has a fidelity acceptable for many users of ERTS data and allows an even greater compaction of data. An improved adaptive form of this algorithm may allow storage of two 100 x 100 n mi scenes on a single reel of tape. The distortion induced at present in the data by the ground decompression algorithms is comparable in degree to that produced by the SSDIM with $|m| = 1$. Proposed techniques for the geometric correction of ERTS imagery involve prediction on interpolation of corrected data intensities, a process which is also essentially information preserving. Instead of cascading the operations of decompression, geometric correction, and data compression the three techniques could be combined into a composite algorithm which would increase processing efficiency while minimizing the data stored.

The SSDI/Rice technique is a viable candidate for spacecraft applications permitting a doubling of the data transmitted to ground. The reconstructed data is of high quality provided the channel bit error rate is 10^{-6} or less. To further decrease the effects of channel errors and allow the use of downlinks having a higher bit error rate, error correction coding can be applied to the compressed data.

In conclusion, the results of this study indicate that the use of data compression on ERTS data yields both economic and operational benefits. The tradeoff between strictly and essentially information preserving forms of the algorithms depends on the effect of slight distortion on the various uses of ERTS data.

5. RECOMMENDATIONS FOR FUTURE WORK

The present study has investigated both the degree and variation in compression for the SSDI algorithms using ground processed ERTS-1 tapes (and one spacecraft data tape). The results obtained in this study are relevant to the objectives set forth in the proposal but as system requirements change in the future it is recommended that similar studies be performed. As future scanners are developed with different resolution, different quantizer, or a different set of spectral bands, the expected level of compression would change. The effects of these different parameters are not completely clear at present but compression should increase as the average spectral band separation is narrowed and as system noise diminishes. Changes in ground processing algorithms can also affect the level of compression. Such system changes should be accompanied by a corresponding re-processing of the data with the same or appropriately modified compression algorithms to ascertain the compression and data statistics that result. It is probably sufficient to re-run only a few selected subscenes for purposes of comparison rather than a study of the present scope.

While one spacecraft data tape was processed and compared to the equivalent ground processed tape during the current study, more conclusive results would be obtained by processing a number of these tapes. Such an investigation would yield a body of data of greater relevance to the use of data compression aboard a spacecraft than is provided by the present study. It is recommended that such a study be limited to simulation of the SSDI and SSDIA algorithms followed by Rice encoding since these appear to be the better candidates for spacecraft applications.

A necessary precursor to the use of data compression aboard spacecraft is a more extensive study of the hardware implementation involved. The preliminary study of an implementation of the SSDI/Rice system, given in Section 3, should be refined and extended to a breadboard model. Such a study would develop a more comprehensive understanding of hardware performance and tradeoffs and permit a detailed comparison of actual with simulated performance, especially in the area of buffer parameters.

Another area in which additional work is required is that of efficient ground-based compression and reconstruction of ERTS data. The computer programs used in the present study were developed for the CDC-6500 digital computer with the goal of obtaining a large number of statistical parameters characterizing the data and algorithm performance. The scope of the current investigation neither required nor permitted an optimization of these algorithms on the basis of ground processing efficiency. Based on the results obtained and the attendant benefits to archiving and tape transmittal, an investigation into the form such an operational system should have appears to be a logical extension of the current study. The appendix describes some preliminary thoughts on a technique, based on the SSDI/Huffman compression algorithm, which permits a rapid processing of ERTS imagery applicable to both large scale computer systems and modest minicomputers.

Although the current investigation concentrated on strictly information preserving techniques, the results obtained with the essentially information preserving SSDIAM algorithm indicate that a slight level of distortion, properly performed, may yield reconstructed imagery acceptable to many users of ERTS data. Such an investigation would determine the level and type of distortion incurred by algorithms such as the SSDIAM and the block-interpolation SSDI and the effects of such processed data on various investigations including both photographic interpreters and computer-aided interpretations of the data. Such a study could lead to an optimized algorithm which would not compromise the intended use of the data anymore than such standard processing techniques as geometric correction, while yielding substantially higher compression than is possible with strictly information preserving techniques.

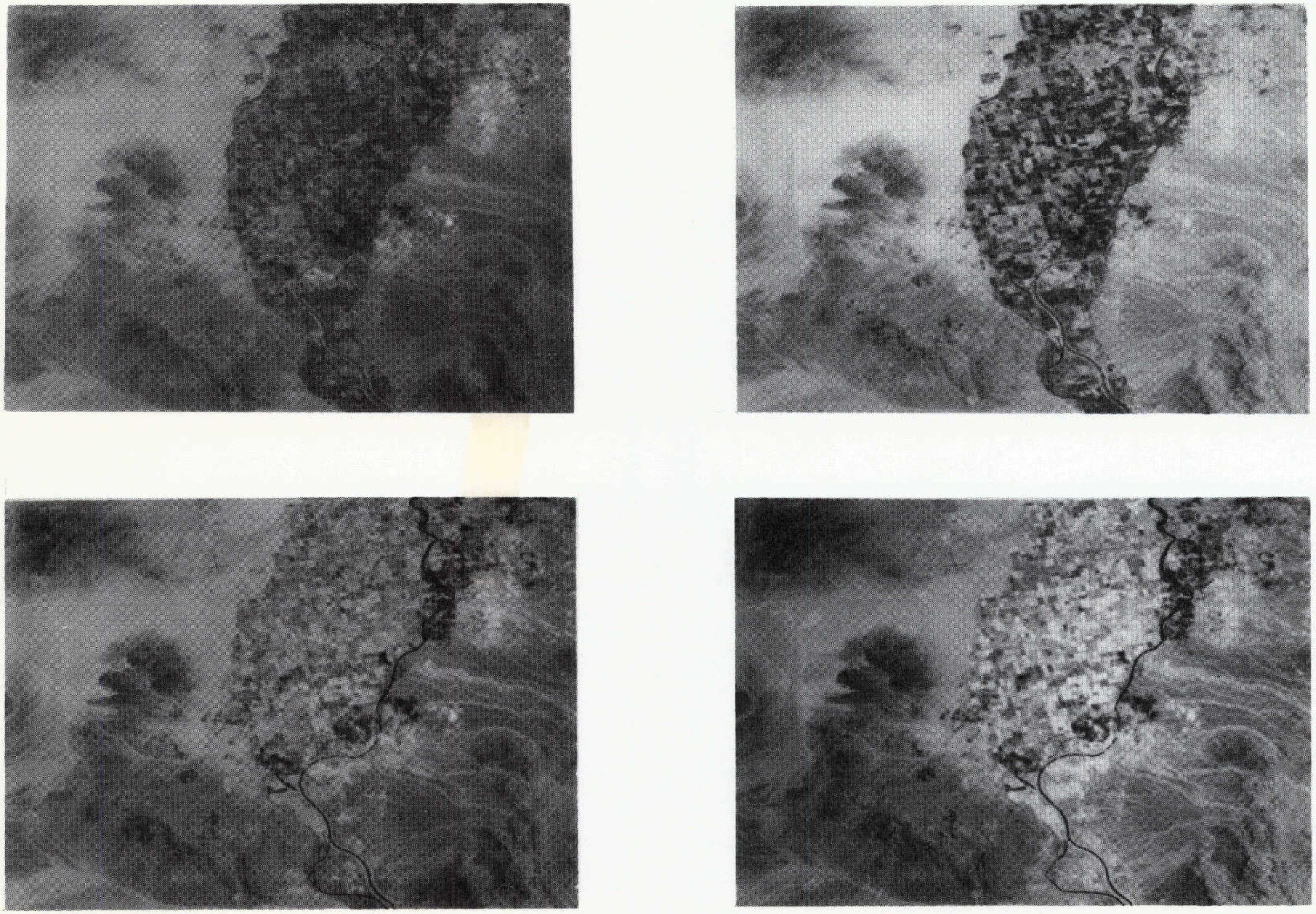
REFERENCES

1. Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes," Proc. IRE, Vol. 40, No. 10, pp. 1098-1101, September 1952.

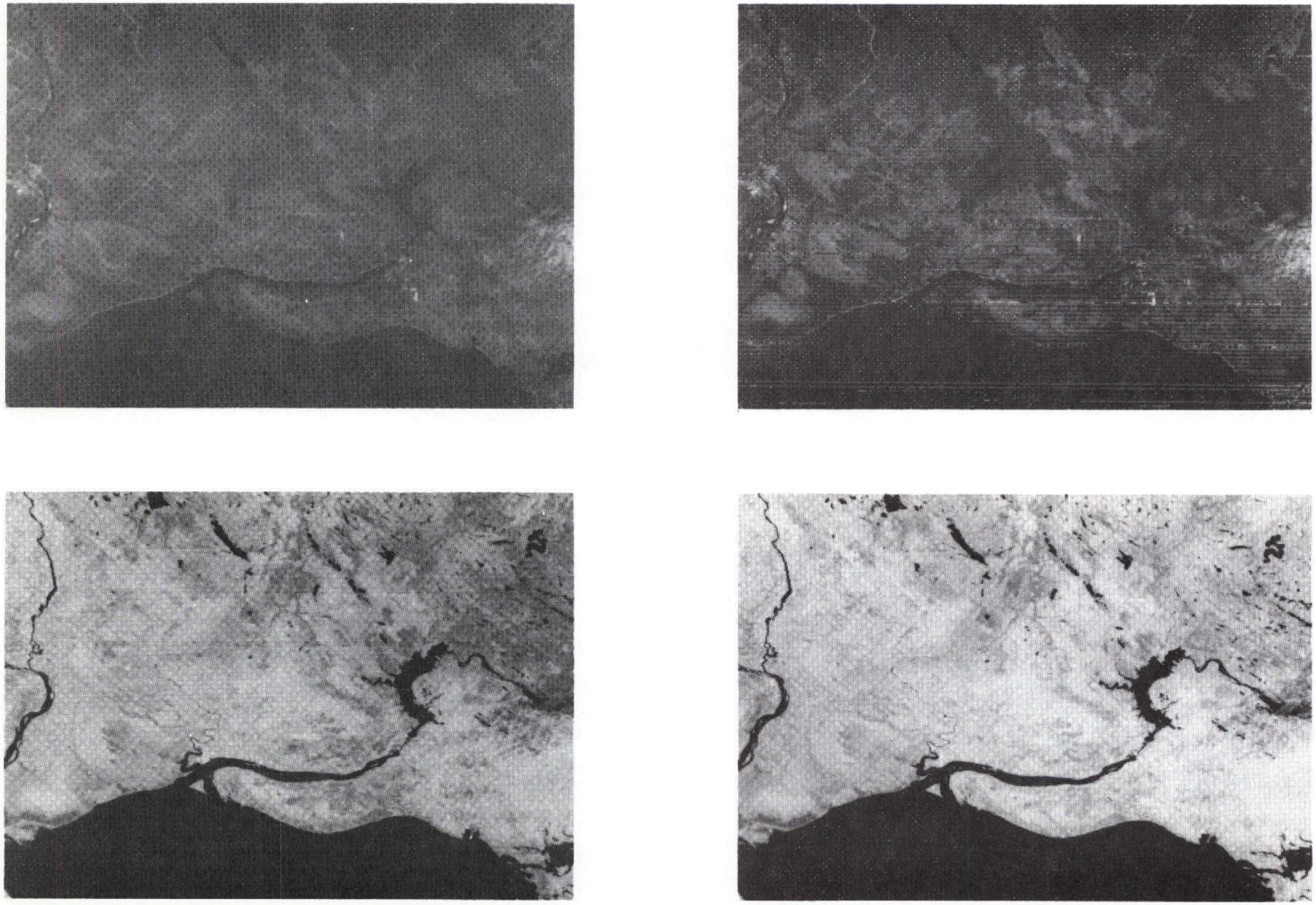
2. Rice, R. F., "The Rice Machine: Television Data Compression," GTD 900-408, JPL, Pasadena, California, September 1970.

APPENDIX A

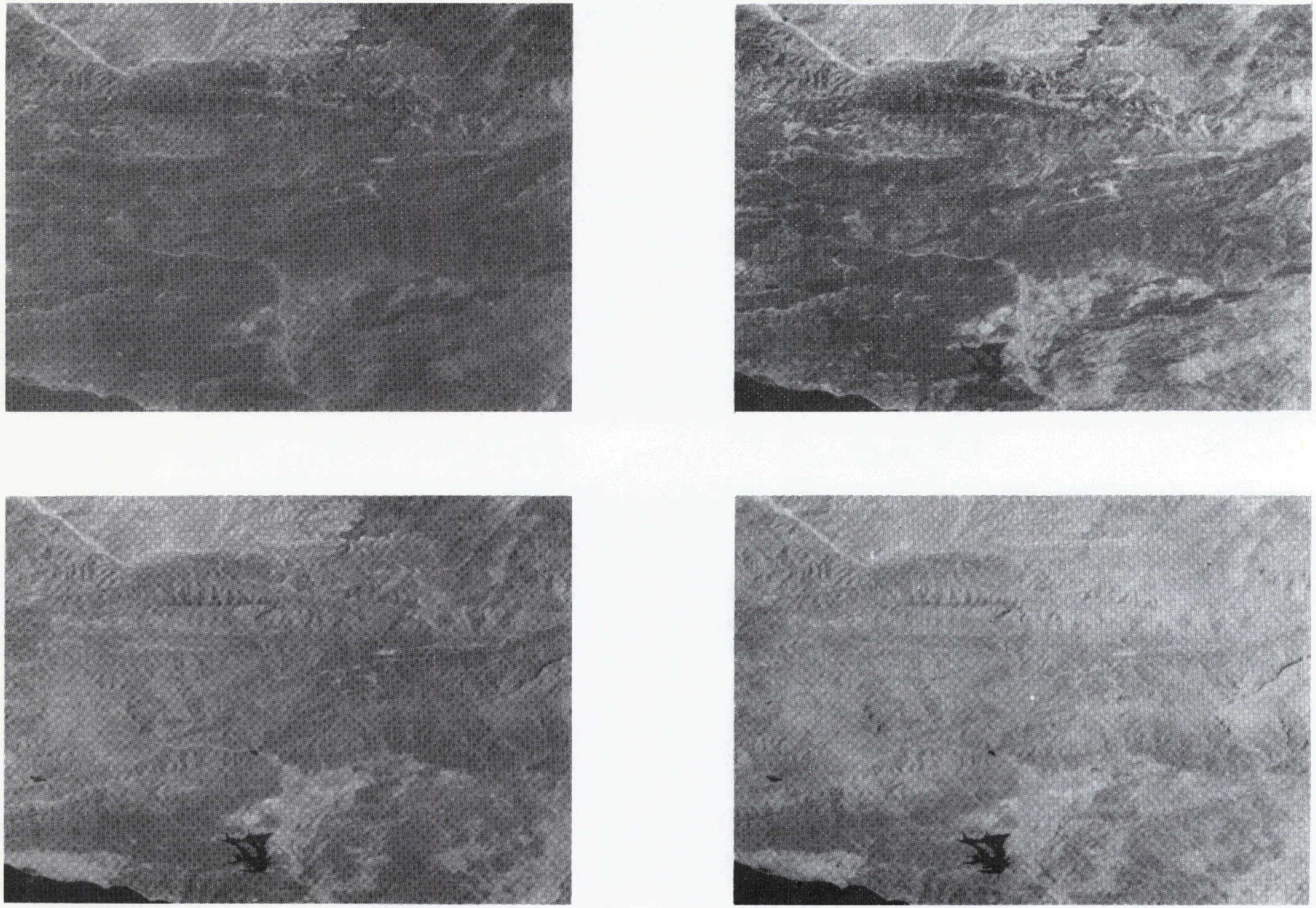
PHOTOS



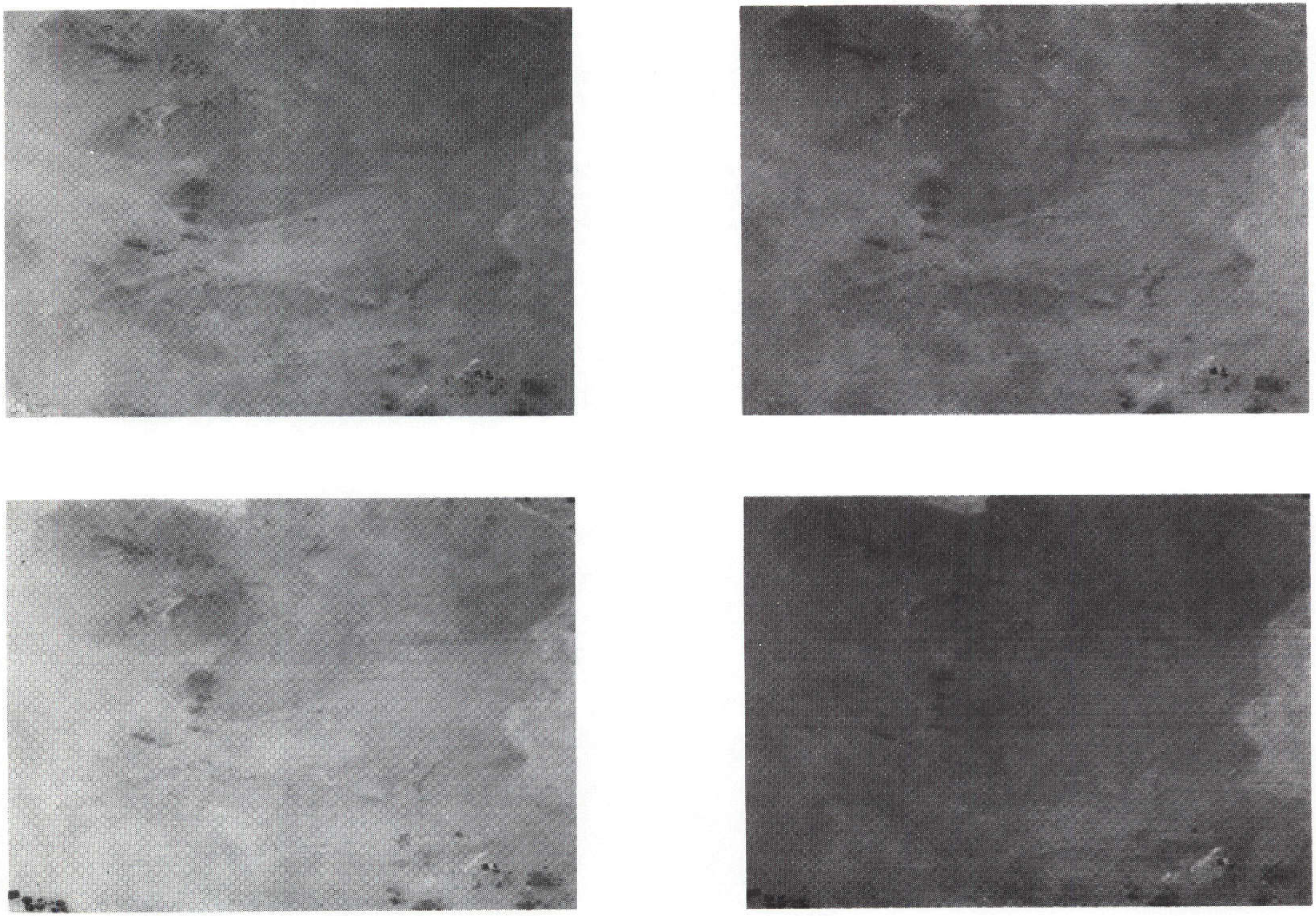
4 Spectral Bands, Processed as Scene 31



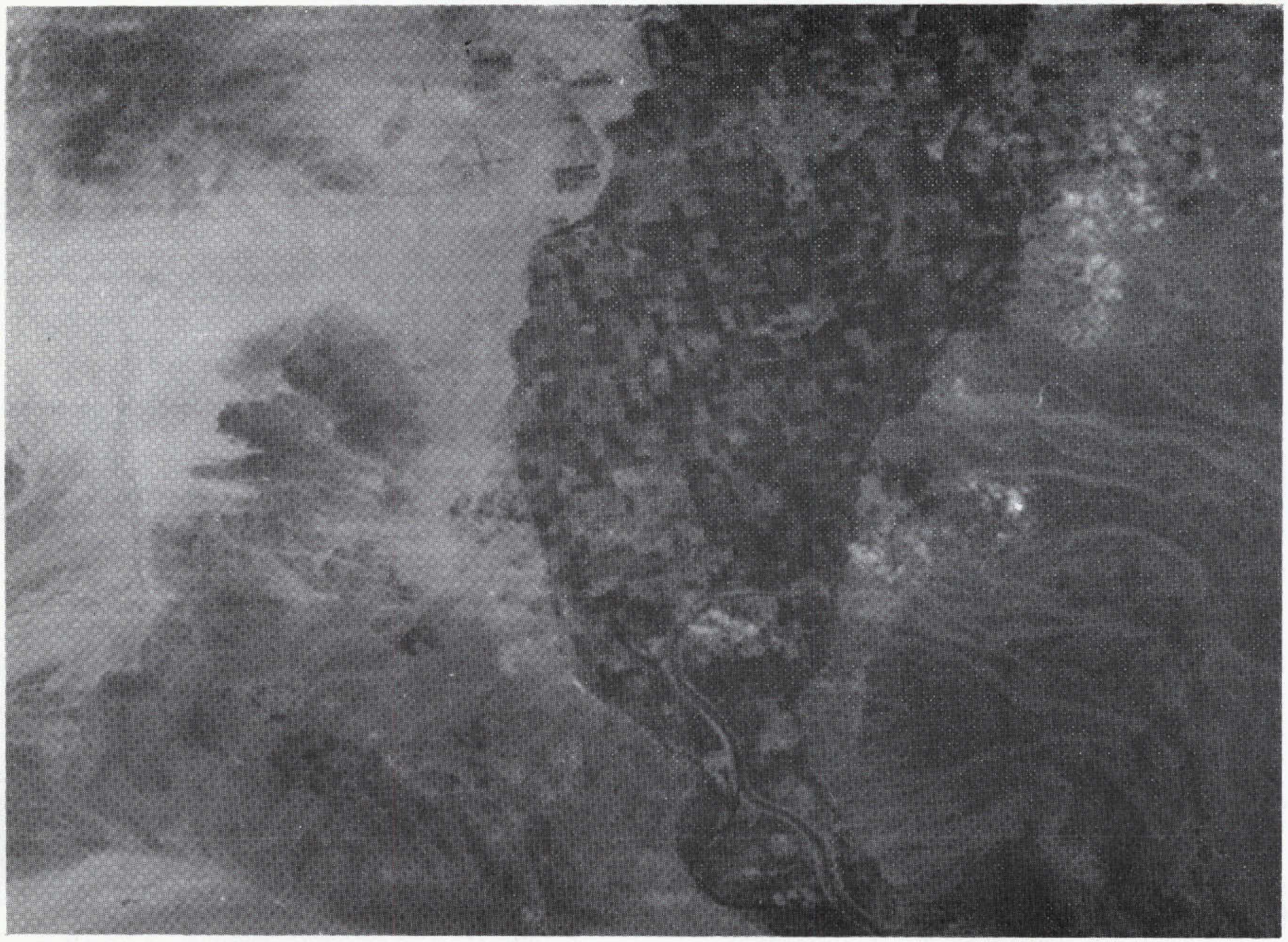
4 Spectral Bands, Processed as Scene 32



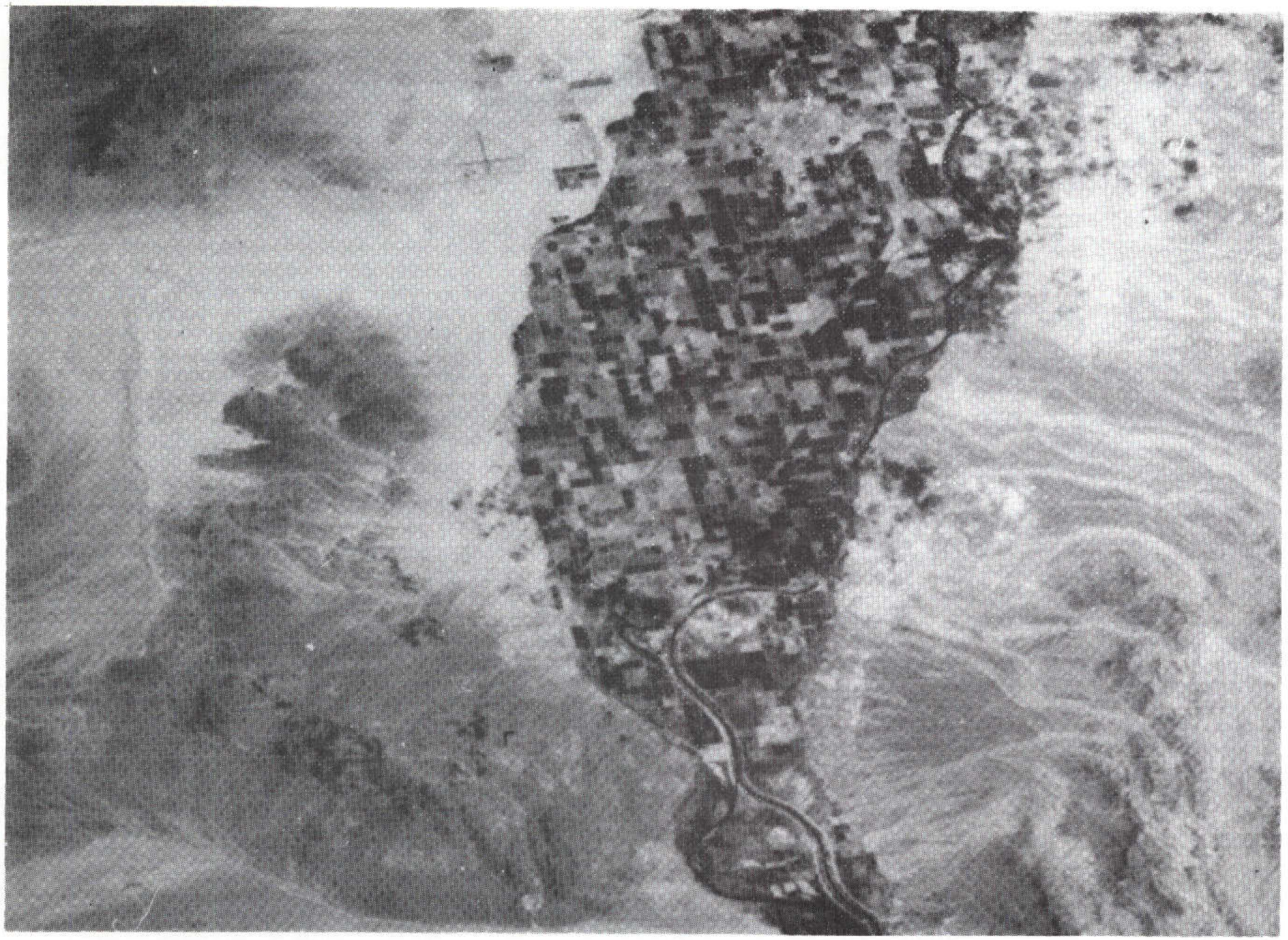
4 Spectral Bands, Processed as Scene 33



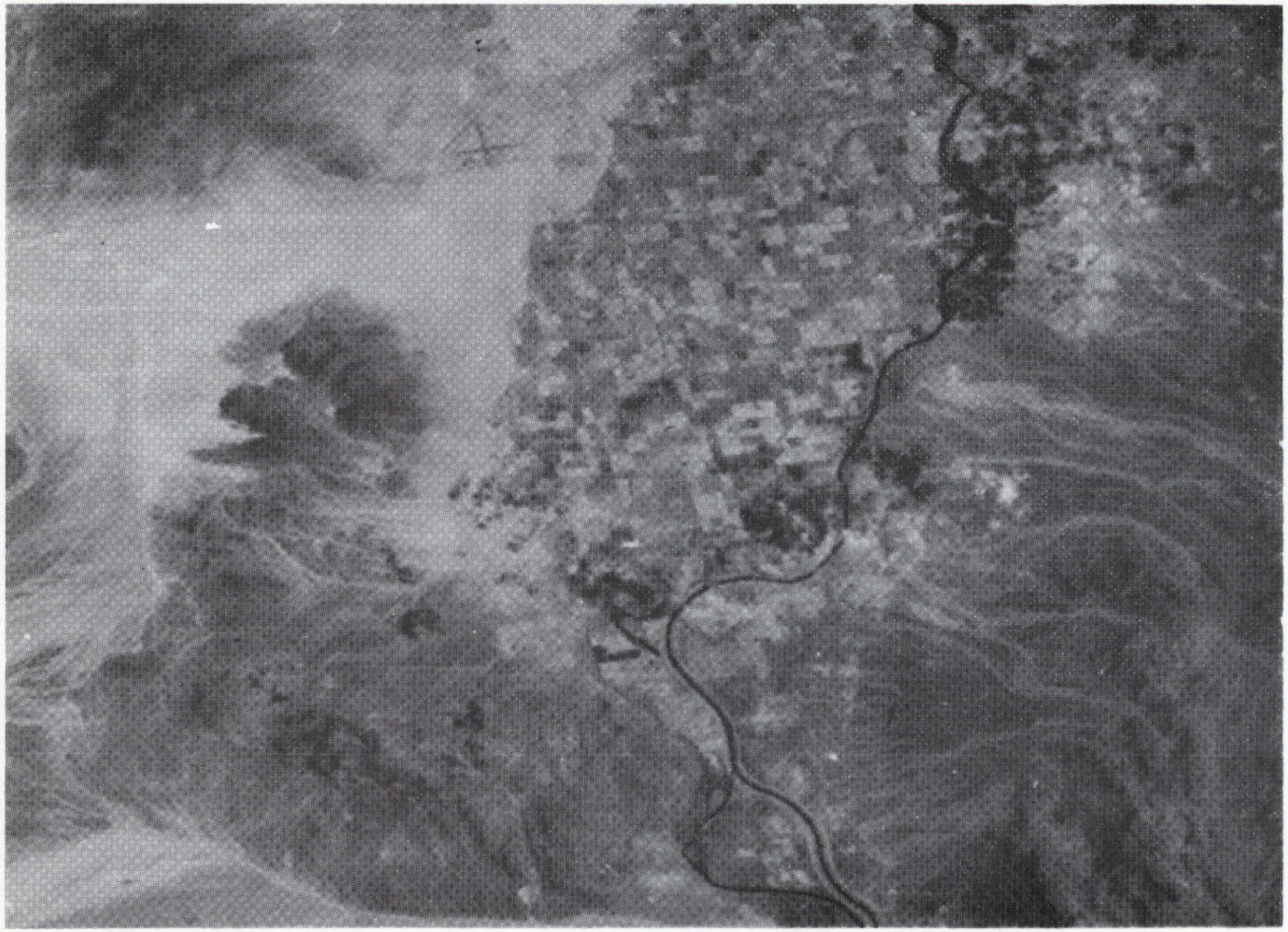
4 Spectral Bands Processed as Scene 34



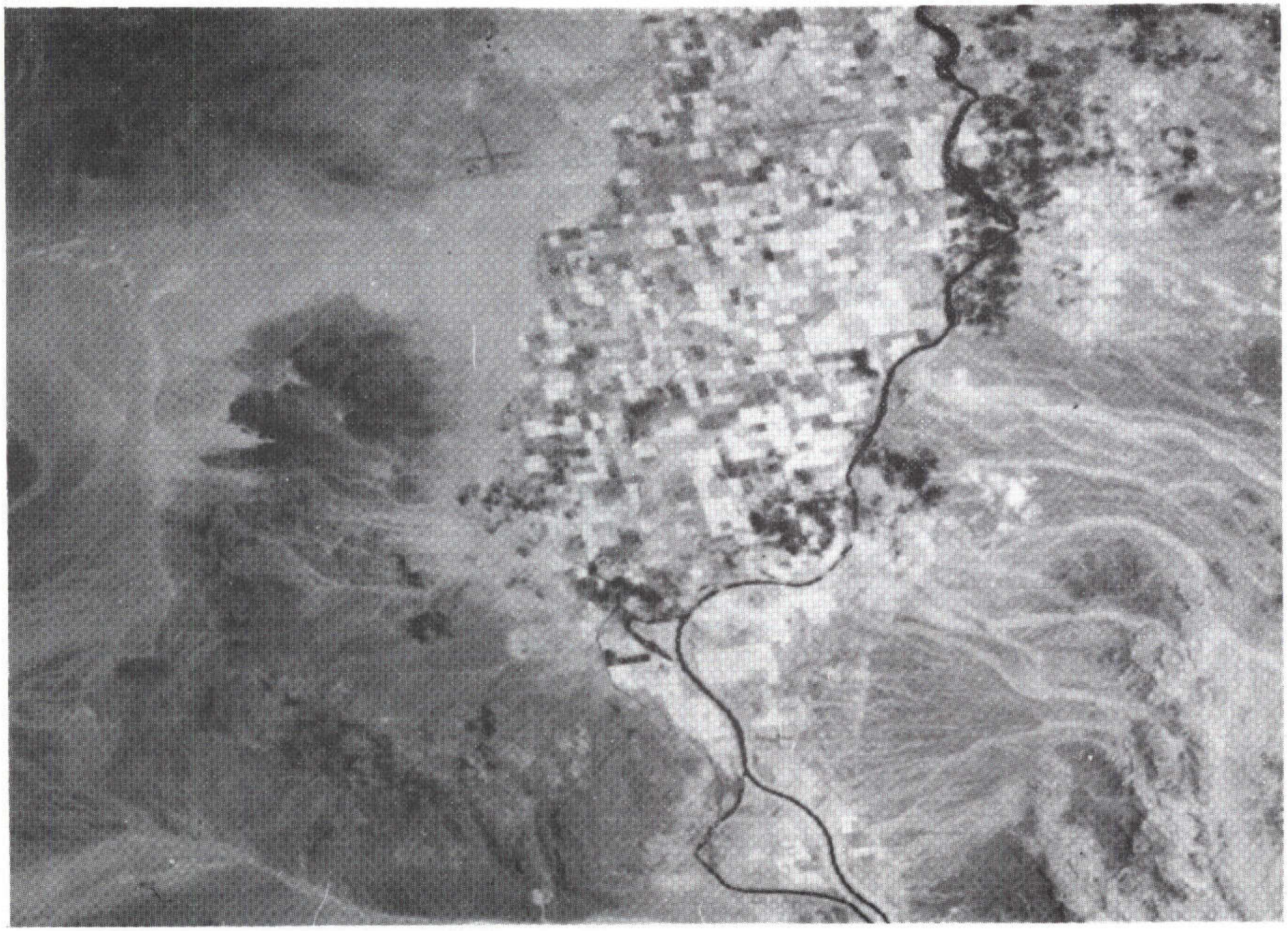
Original Data (Scene 31), Band 1



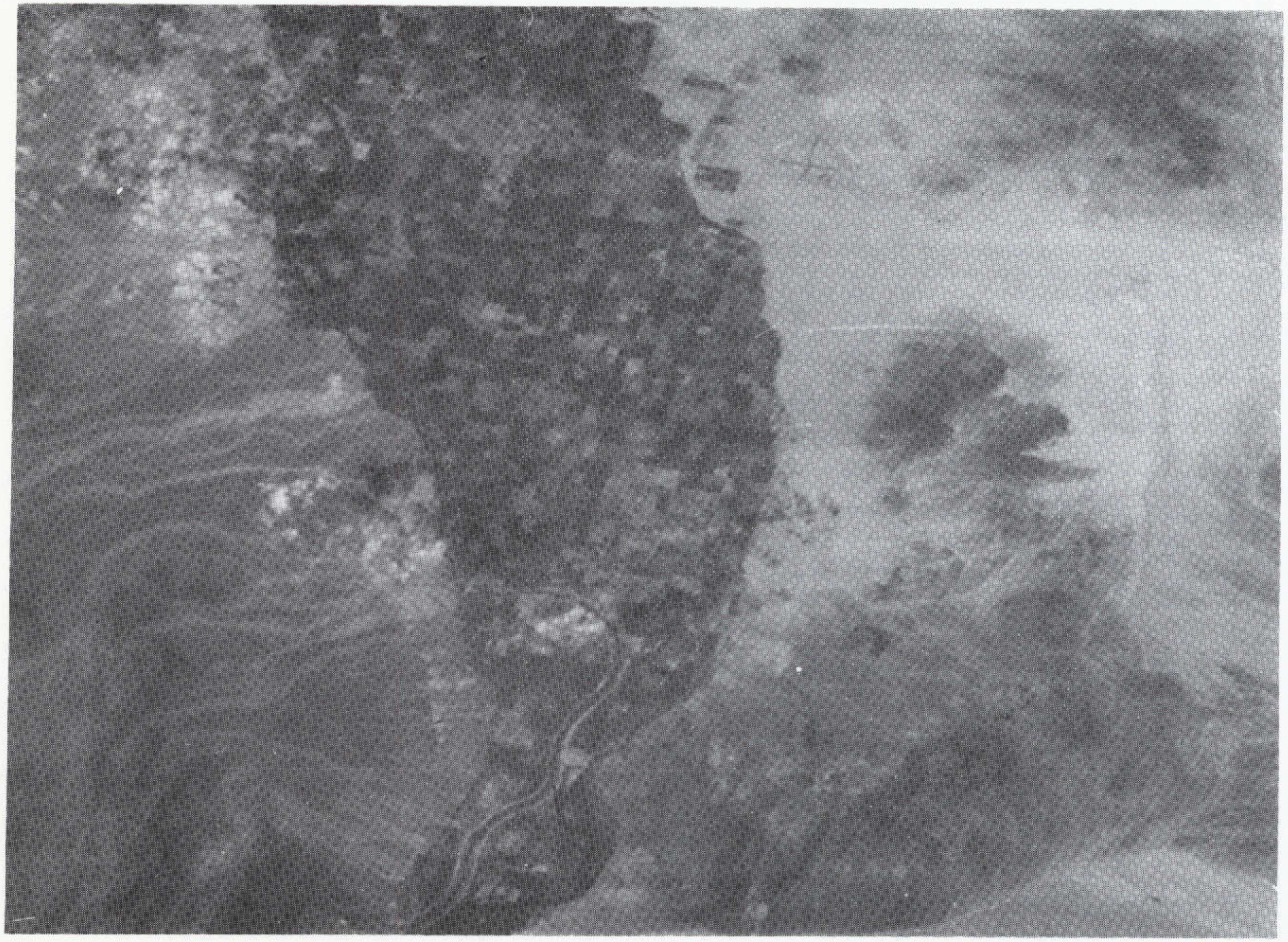
Original Data (Scene 31), Band 2



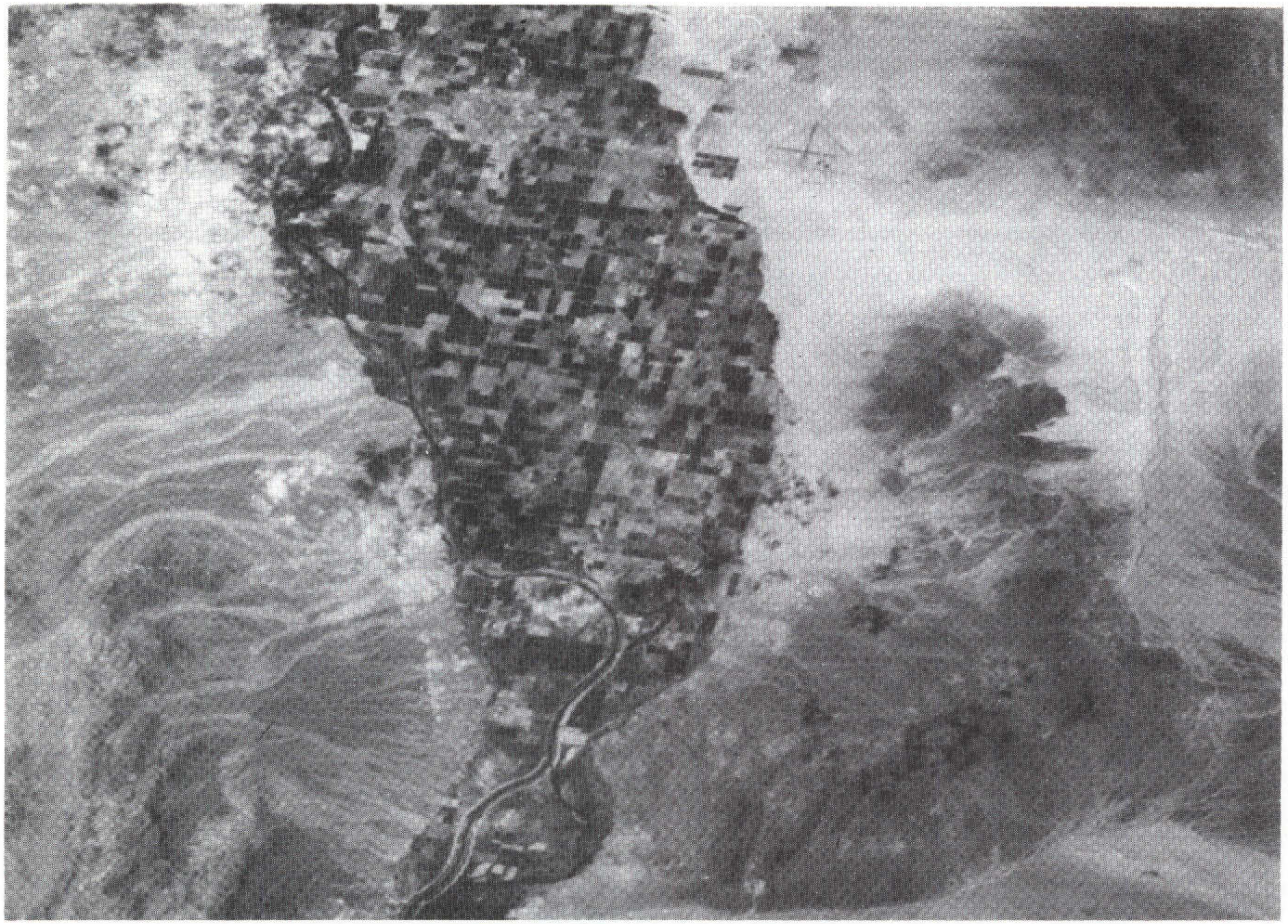
Original Data (Scene 31), Band 3



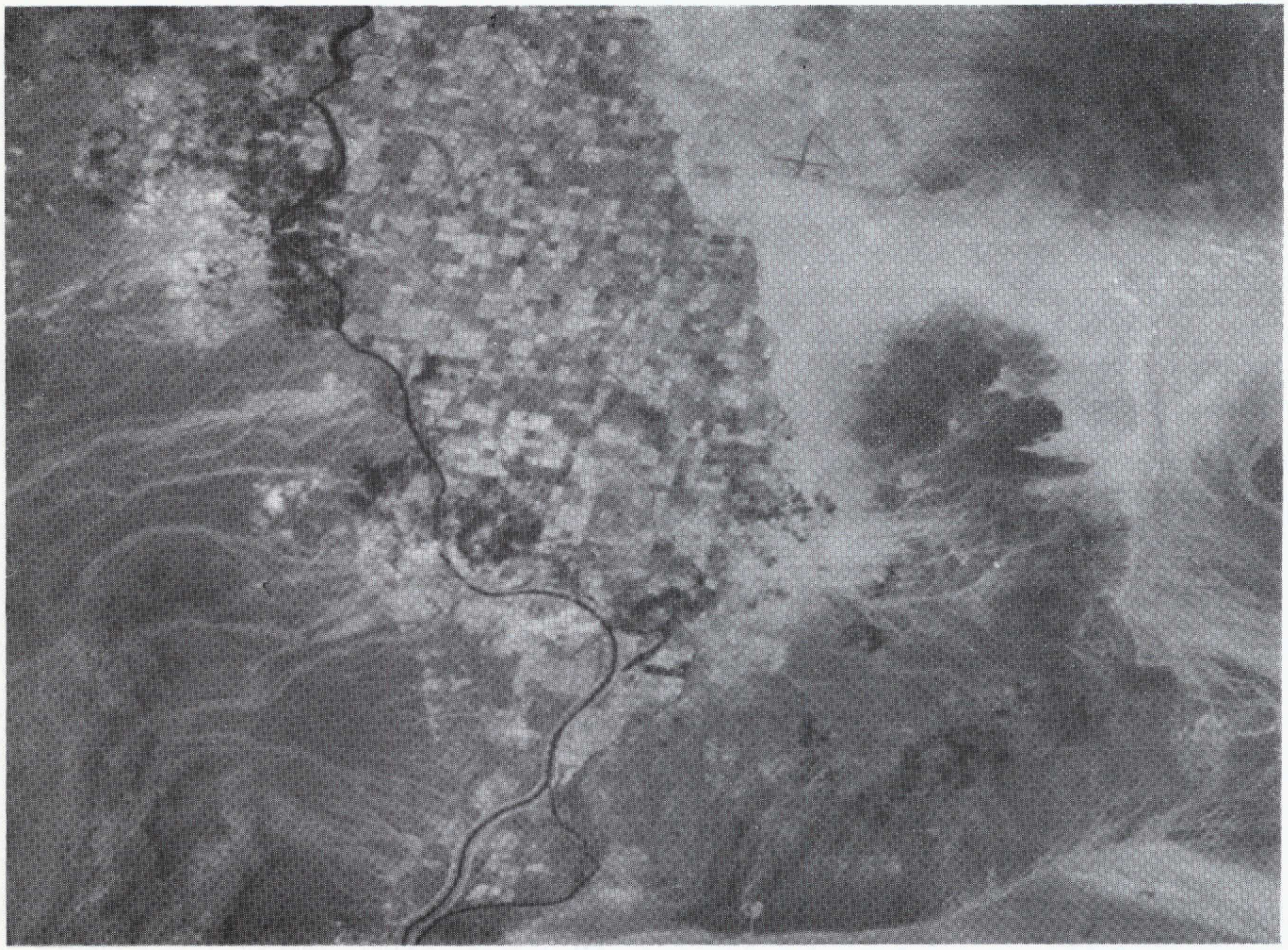
Original Data (Scene 31), Band 4



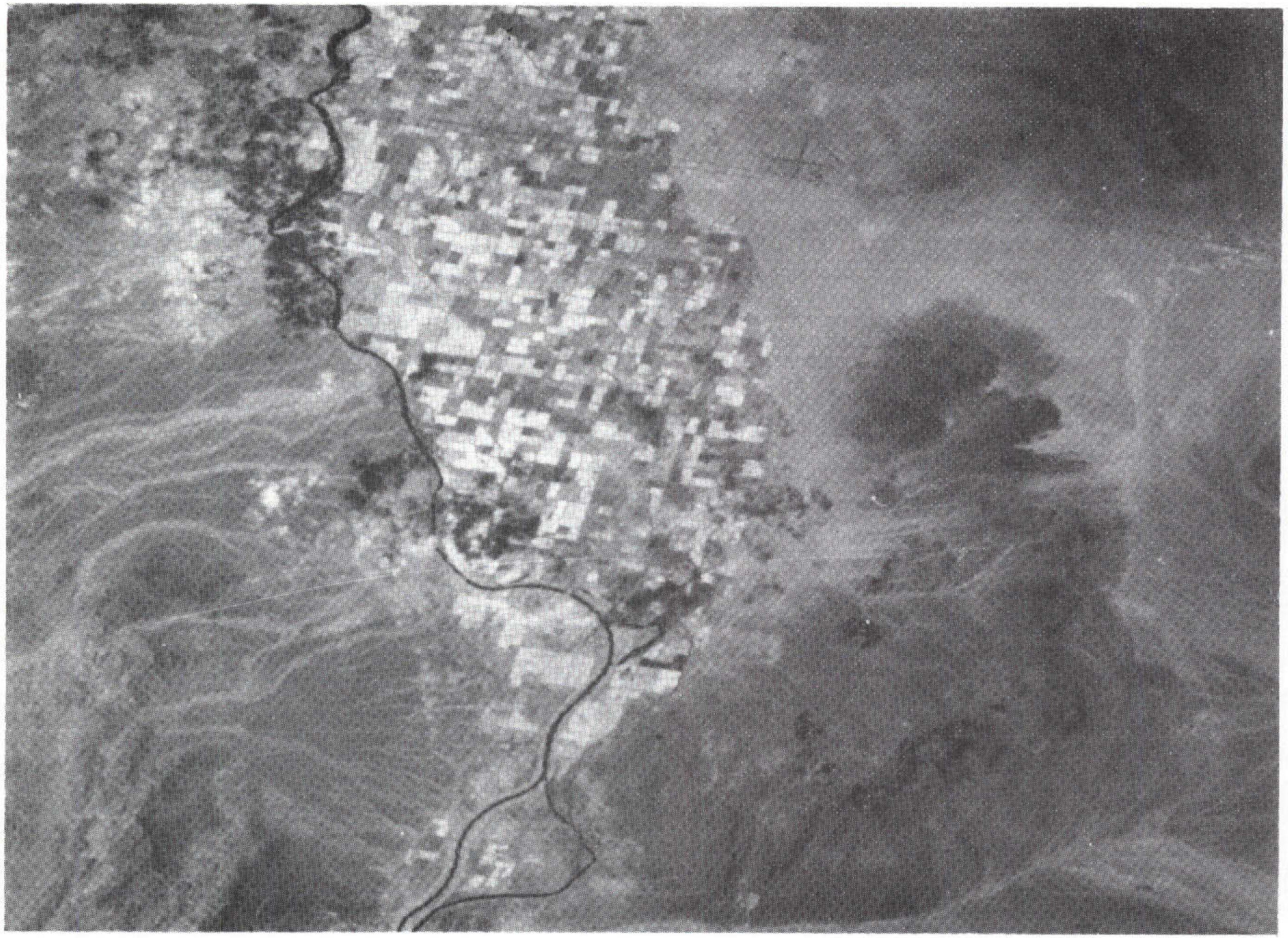
Essentially Information Preserving Data, |m|=1 (Scene 31), Band 1



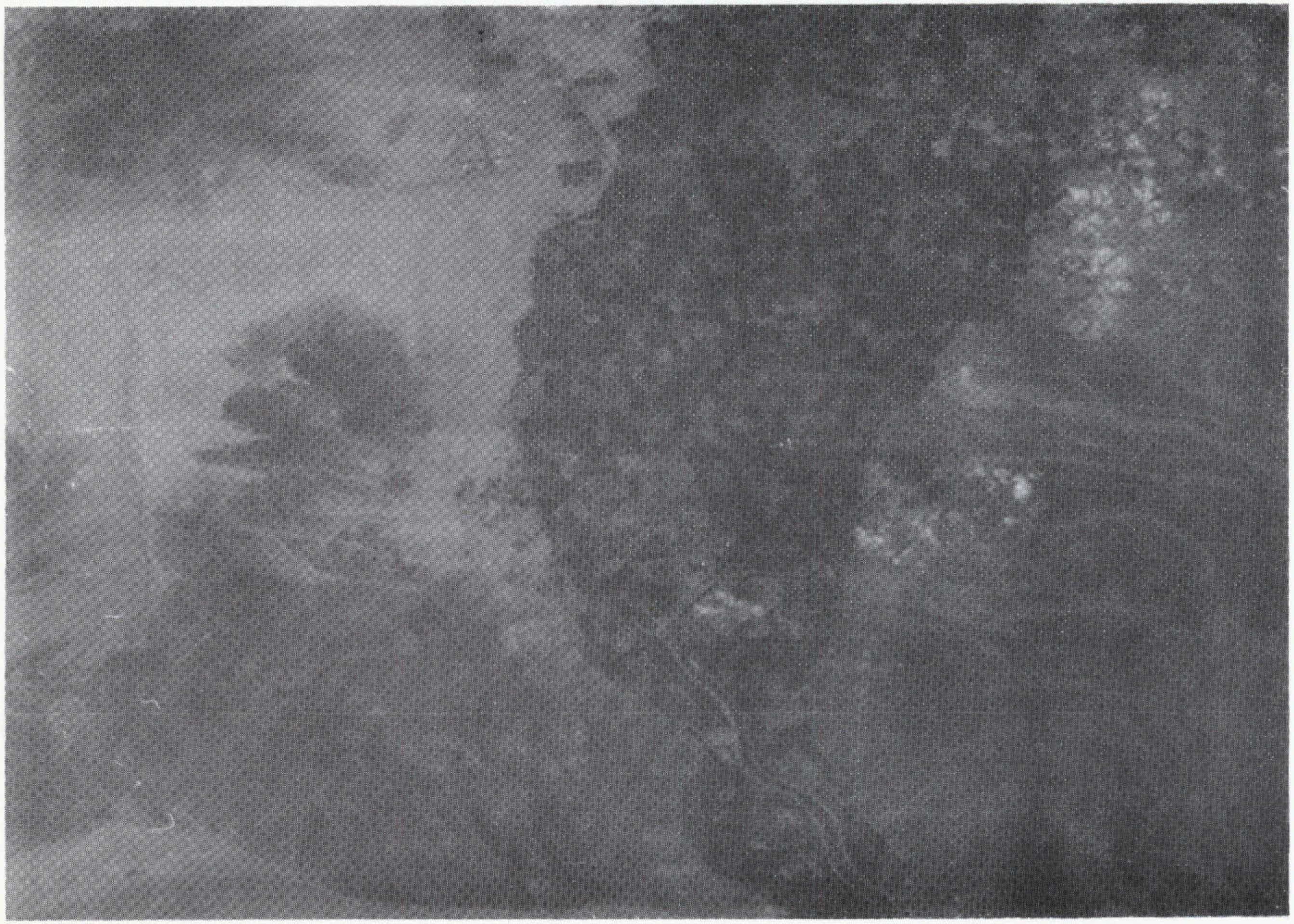
Essentially Information Preserving Data, |m|=1 (Scene 31), Band 2



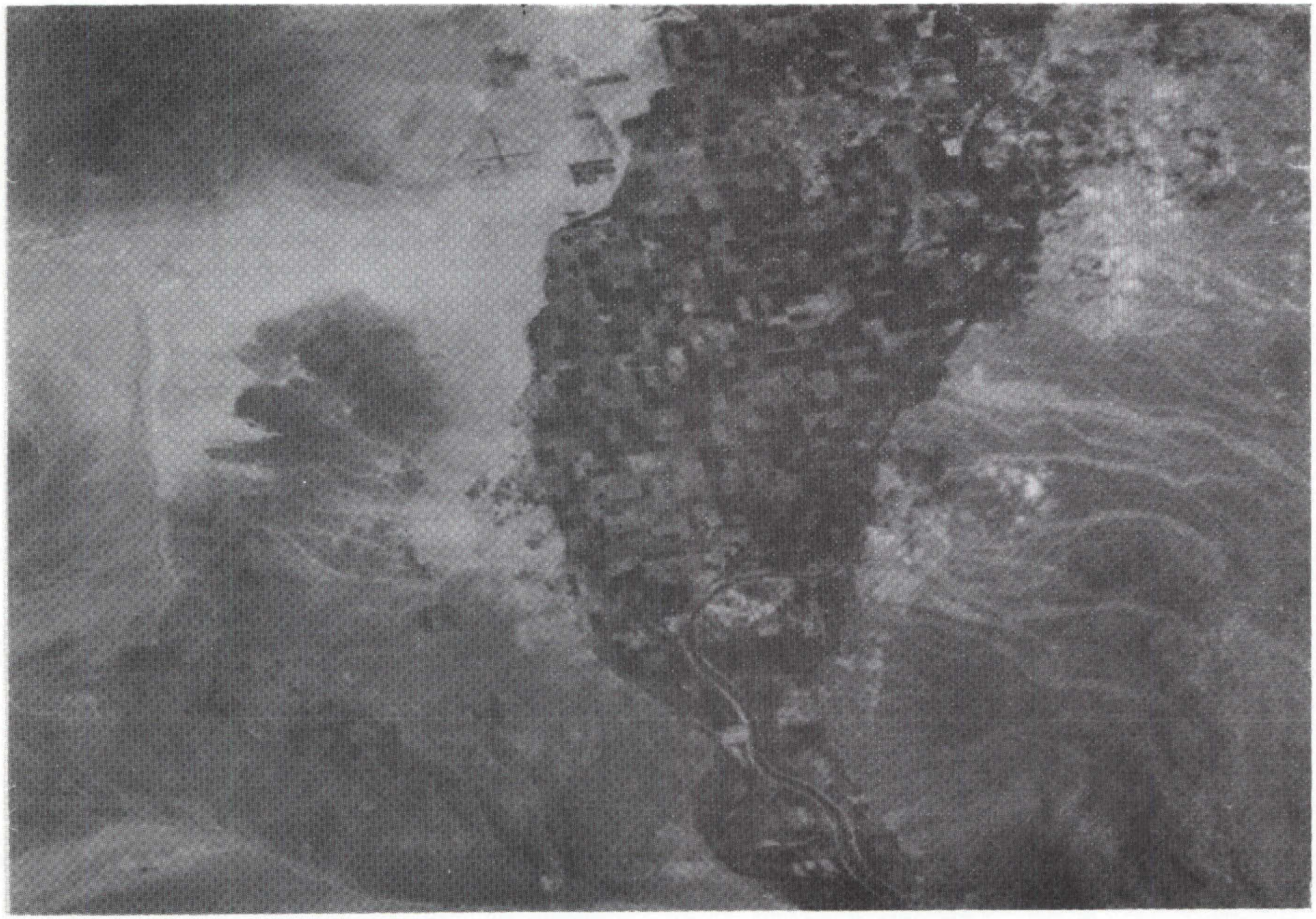
Essentially Information Preserving Data, m =1 (Scene 31), Band 3



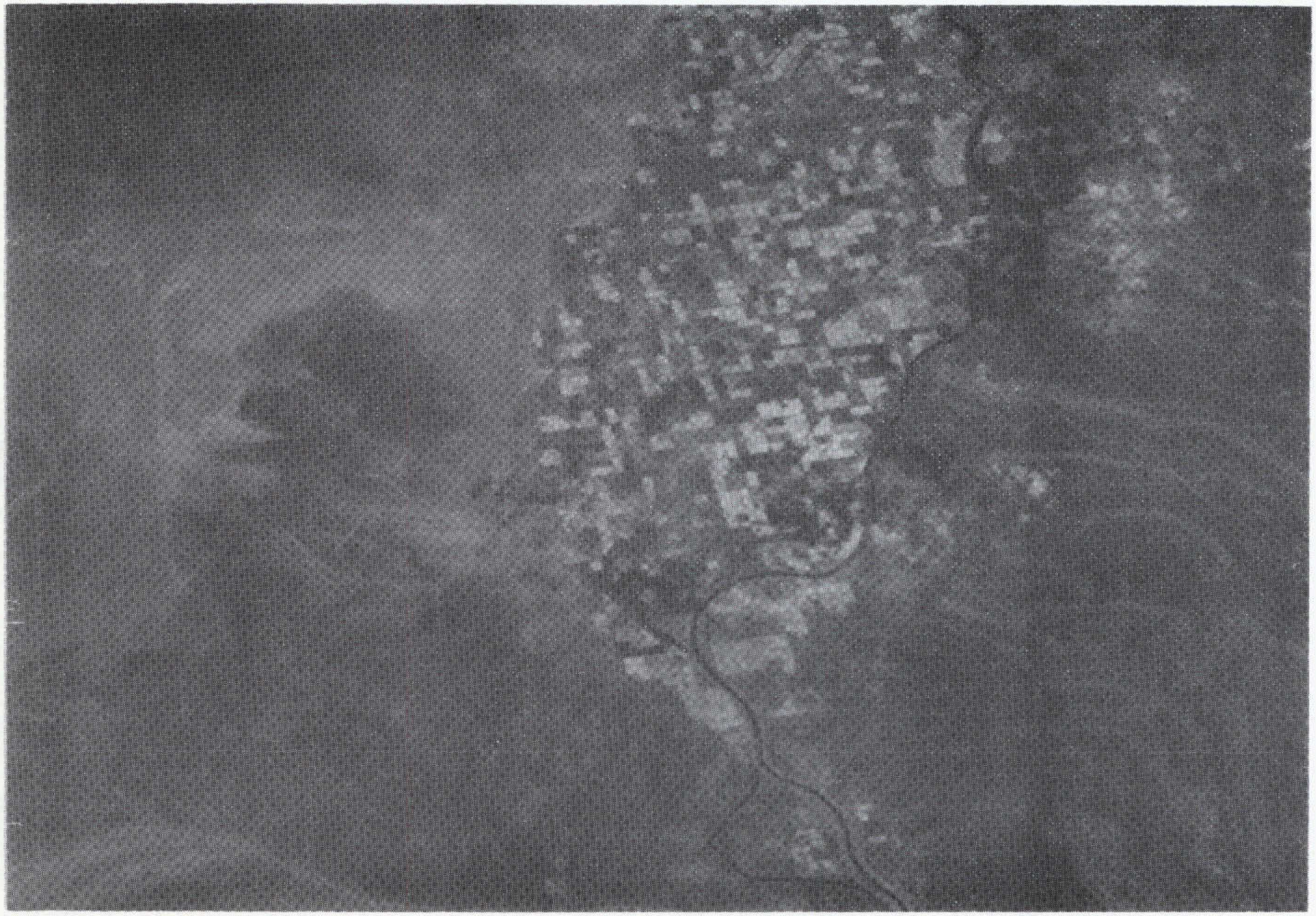
Essentially Information Preserving Data, /m|=1 (Scene 31), Band 4



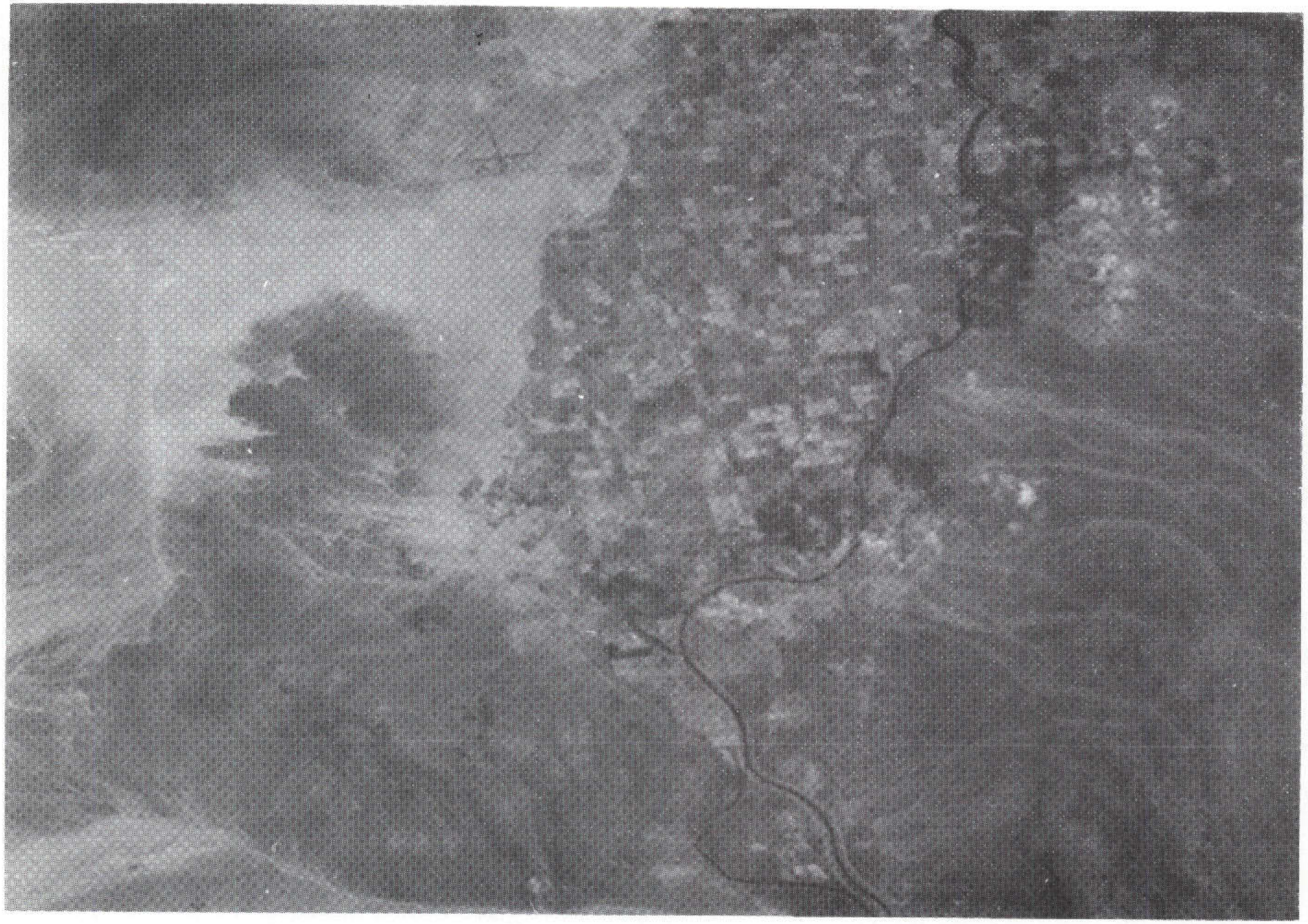
Essentially Information Preserving Data, $|m|=3$ (Scene 31), Band 1



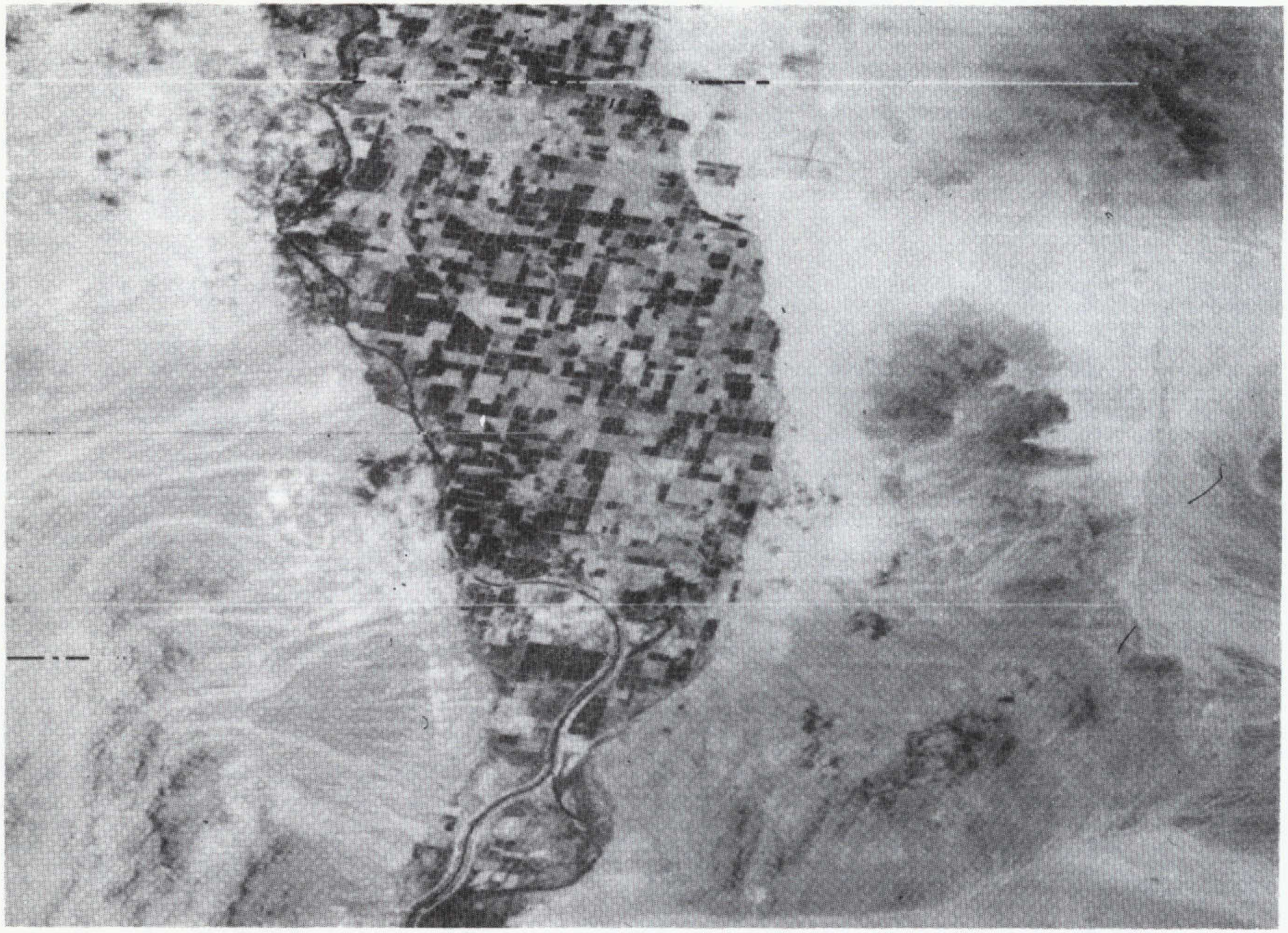
Essentially Information Preserving Data, $|m|=3$ (Scene 31), Band 2



Essentially Information Preserving Data, $|m|=3$ (Scene 31), Band 3



Essentially Information Preserving Data, m =3 (Scene 31), Band 4



Reconstructed Data with Channel Bit Error Rate = 10^{-5}, Band 2

APPENDIX B: COMPUTER OUTPUT

This section contains the set of computer print output resulting from the operation of programs DCSTAT1 and DCSTAT2 on the Principal Full Scene (number 31).

CUPPELATICN

STEPSIZE PERCENT	1 PROB	2 PROB	4 PROB	6 PROB	8 PROB
100.000	0.000	0.000	0.000	0.000	0.000
99.000	.954	.855	.795	.781	.774
98.000	.985	.939	.895	.879	.871
97.000	.993	.966	.935	.920	.913
96.000	.996	.979	.955	.943	.937
95.000	.997	.986	.968	.958	.953
94.000	.998	.990	.977	.969	.965
93.000	.999	.992	.983	.977	.974
92.000	.999	.994	.987	.982	.980
91.000	.999	.995	.990	.987	.986
90.000	1.000	.996	.992	.990	.989
89.000	1.000	.997	.994	.993	.992
88.000	1.000	.997	.995	.995	.995
87.000	1.000	.998	.996	.996	.996
86.000	1.000	.998	.996	.997	.997
85.000	1.000	.999	.997	.997	.998
84.000	1.000	.999	.997	.998	.998
83.000	1.000	.999	.998	.998	.998
82.000	1.000	.999	.998	.999	.999
81.000	1.000	.999	.999	.999	.999
80.000	1.000	1.000	.999	.999	.999
79.000	1.000	1.000	.999	.999	.999
78.000	1.000	1.000	.999	1.000	1.000
77.000	1.000	1.000	.999	1.000	1.000
76.000	1.000	1.000	1.000	1.000	1.000
75.000	1.000	1.000	1.000	1.000	1.000
74.000	1.000	1.000	1.000	1.000	1.000
73.000	1.000	1.000	1.000	1.000	1.000
72.000	1.000	1.000	1.000	1.000	1.000
71.000	1.000	1.000	1.000	1.000	1.000

	BAND1	BAND2	BAND3	BAND4	TOTAL
MU	51.220	57.761	56.890	23.880	47.438
SIGMA	174.436	273.124	170.609	43.519	356.721

JOINT PROBABILITY ELLIPSE - BAND 1 .VS. BAND 2

```
    -16-15-14-13-12-11-10 -9 -8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
 -16
 -15
 -14
 -13
 -12
 -11
 -10
  -9
  -8
  -7                                                  1
  -6                                                  1
  -5                                            2     5     3  1  2     1
  -4                                         1  1     5     2  1  1
  -3                                1     2  3  5  1 16  1  6  4  3  2  2
  -2                                2  2  3  4  7  1 23  2  9  5  5  2  2  1
  -1                                            2     4     2
   0                      1  1  2  3  6  5 13 21 24  5100  5 24 21 13  6  6  3  2  1  1
   1                                            2     4     1
   2                               1  2  2  5  5  8  2 24  1  7  4  3  2  2
   3                                2  2  3  4  5    16     4  3  2     1
   4                                      1  2  2     5     1  1
   5                                1     2  1  2     5     1  1
   6                                                  1
   7                                                  1
   8
   9
  10
  11
  12
  13
  14
  15
  16
```

```
                 JOINT PROBABILITY ELLIPSE - BAND 1 .VS. BAND 3

    -16-15-14-13-12-11-10 -9 -8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16

 -16
 -15
 -14
 -13
 -12
 -11
 -10
  -9
  -8
  -7                                                  2
  -6                                                  2
  -5                                       2  2     6     3  2     1
  -4                                       2  2     6     2  2
  -3                              1  2  2  5  6    17     6  6  2  2  1
  -2                              2  3  3  7  9  1 25  1 10  7  3  3  2  1
  -1                                       1  2     5     2  1
   0                       2  2  3  6 10  8 32 30  3100  3 31 32  8 10  6  3  2  2
   1                                       1  2     5     2  1
   2                              2  3  3  8  9  1 25  1  9  8  2  3  2
   3                              1  2  2  6  6    17     6  5  2  2  1
   4                                       2  2     6     2  2
   5                                 1     2  2     6     2  2
   6                                                  2
   7                                                  2
   8
   9
  10
  11
  12
  13
  14
  15
  16
```

JOINT PROBABILITY ELLIPSE - BAND 1 .VS. BAND 4

	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-16																																	
-15																																	
-14																																	
-13																																	
-12																																	
-11																																	
-10																																	
-9																																	
-8																																	
-7																1	2	1															
-6																1	2	1															
-5															2	4	6	4	2														
-4															1	3	6	3	1														
-3														2	4	10	17	11	4	2													
-2													1	3	6	16	25	16	7	3	1												
-1															1	3	4	3	2														
0												2	3	8	21	60	100	59	21	8	4	2											
1															1	3	5	3	2														
2													1	2	7	16	25	16	7	3	1												
3														2	5	11	17	11	4	2													
4															1	4	6	3	1														
5															2	4	6	4	2														
6																1	2	1															
7																1	2	1															
8																																	
9																																	
10																																	
11																																	
12																																	
13																																	
14																																	
15																																	
16																																	

JOINT PROBABILITY ELLIPSE - BAND 2 .VS. BAND 3

	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-16																																	
-15																																	
-14																																	
-13																																	
-12																																	
-11																																	
-10																	1																
-9																	2																
-8																	2																
-7																	3		1	1													
-6														2	2		6		3	2	1	1											
-5														1	2		6		3	2	1	1											
-4												1	1	3	4		13		6	4	2	2	1										
-3											1	2		8	5		22		5	8	1	2	2										
-2											1	2	3	5	9	2	23	2	12	6	4	3	2	1									
-1														1	2		5		2	2													
0								1	2	2	6	9	7	32	29	3	100	3	29	31	7	9	5	2	2	1							
1														1	2		5		2	1													
2										1	2	3	4	6	10	1	24	1	10	6	3	3	1	1									
3											2	3	1	9	5		22		4	7		1	1										
4											1	2	2	4	5		13		5	4	1	2											
5												1		2	2		6		2	2													
6												1		3	2		6		2	2													
7														1	1		3			1													
8																	2																
9																	2																
10																	1																
11																																	
12																																	
13																																	
14																																	
15																																	
16																																	

JOINT PROBABILITY ELLIPSE - BAND 2 .VS. BAND 4

	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-16																																	
-15																																	
-14																																	
-13																																	
-12																																	
-11																																	
-10																	1																
-9																1	1	1															
-8																1	2	1															
-7															1	2	3	2	1														
-6															2	4	6	5	2	1													
-5															2	4	6	4	2														
-4														1	3	8	13	9	4	2													
-3														2	4	13	21	12	4	2													
-2													1	2	6	15	24	17	7	3	1												
-1															1	3	5	3	2														
0												1	3	7	20	59	100	57	19	7	3	2											
1															2	3	5	3	1														
2													1	3	7	16	25	17	7	3	1												
3														2	5	13	21	12	4	1													
4														2	4	9	13	9	4	1													
5															2	4	6	4	2														
6														1	2	5	6	4	2														
7															1	2	3	2	1														
8																1	2	1															
9																1	2	1															
10																	1																
11																																	
12																																	
13																																	
14																																	
15																																	
16																																	

JOINT PROBABILITY ELLIPSE - BAND 3 .VS. BAND 4

	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-16																																	
-15																																	
-14																																	
-13																																	
-12																																	
-11																																	
-10																																	
-9																	1	1															
-8																1	2	2	1														
-7																1	2	2	1														
-6															1	3	5	4	2	1													
-5															3	5	9	7	4	2													
-4															1	4	7	6	4	2													
-3													1	2	6	18	31	18	7	2	1												
-2														2	6	18	30	21	9	3	1												
-1																2	3	3	1														
0												1	3	7	19	59	100	57	19	7	3	1											
1																2	3	2	1														
2													1	3	8	20	31	19	7	3	1												
3													2	3	8	20	30	16	5	2													
4														1	3	6	7	5	2	1													
5														2	4	7	9	5	2														
6														1	2	5	5	3	1														
7															1	2	2	2															
8															1	2	2	1															
9																1	1																
10																																	
11																																	
12																																	
13																																	
14																																	
15																																	
16																																	

OVERALL PROBABILITY

LEVEL	SSDI	SSDIA	SSDIAM
-18	.00	.00	.00
-17	.00	.00	.00
-16	.00	.00	.00
-15	.00	.00	.00
-14	.00	.00	.00
-13	.00	.00	.00
-12	.00	.00	.00
-11	.00	.00	.00
-10	.00	.00	.00
-9	.00	.00	.00
-8	.01	.01	.00
-7	.01	.01	.00
-6	.02	.01	.00
-5	.02	.02	.01
-4	.03	.03	.01
-3	.07	.05	.03
-2	.10	.08	.07
-1	.06	.12	.17
0	.34	.30	.32
1	.06	.12	.22
2	.10	.08	.10
3	.07	.05	.04
4	.03	.03	.02
5	.02	.02	.01
6	.02	.01	.00
7	.01	.01	.00
8	.01	.01	.00
9	.00	.00	.00
10	.00	.00	.00
11	.00	.00	.00
12	.00	.00	.00
13	.00	.00	.00
14	.00	.00	.00
15	.00	.00	.00
16	.00	.00	.00
17	.00	.00	.00
18	.00	.00	.00

HUFFMAN CODES FOR SHELL

AVERAGE CODE LENGTH 3.739

LEVEL	PROB.	LENGTH	
1	.023	6	111100
2	.033	5	11100
3	.151	3	011
4	.230	2	00
5	.132	3	010
6	.122	3	101
7	.089	3	100
8	.055	4	1100
9	.039	5	11011
10	.031	5	11010
11	.022	6	111011
12	.015	6	111101
13	.014	7	1111101
14	.009	7	1111100
15	.007	7	1111110
16	.005	8	11111111
17	.005	8	11111110
18	.003	6*	111010
19	.003	6*	111010
20	.002	6*	111010
21	.002	6*	111010
22	.001	6*	111010
23	.001	6*	111010
24	.001	6*	111010
25	.001	6*	111010

*GROUPED PROBABILITY = .018 CODE LENGTH = 6

HUFFMAN CODES FOR SSDI

AVERAGE CODE LENGTH 3.686 ENTROPY 3.507

LEVEL	PROB.	LENGTH	
-20	.000	4*	1000
-19	.000	4*	1000
-18	.000	4*	1000
-17	.000	4*	1000
-16	.001	4*	1000
-15	.001	4*	1000
-14	.001	4*	1000
-13	.001	4*	1000
-12	.002	4*	1000
-11	.002	4*	1000
-10	.003	4*	1000
-9	.005	4*	1000
-8	.006	7	1111111
-7	.009	7	1111110
-6	.016	6	111101
-5	.025	5	11100
-4	.030	5	11011
-3	.069	4	1100
-2	.096	3	011
-1	.063	4	1011
0	.337	2	00
1	.063	4	1010
2	.097	3	010
3	.069	4	1001
4	.030	5	11010
5	.025	5	11101
6	.016	6	111100
7	.009	7	1111101
8	.006	7	1111100
9	.005	4*	1000
10	.003	4*	1000
11	.002	4*	1000
12	.002	4*	1000
13	.001	4*	1000
14	.001	4*	1000
15	.001	4*	1000

```
  16      .001          4*      1000
  17      .000          4*      1000
  18      .000          4*      1000
  19      .000          4*      1000
  20      .000          4*      1000

 *GROUPED PROBABILITY =   .036   CODE LENGTH =   4
```

HUFFMAN CODES FOR SSDIA

AVERAGE CODE LENGTH 3.696 ENTROPY 3.504

LEVEL	PROB.	LENGTH	
-20	.000	4*	1000
-19	.000	4*	1000
-18	.000	4*	1000
-17	.000	4*	1000
-16	.000	4*	1000
-15	.001	4*	1000
-14	.001	4*	1000
-13	.001	4*	1000
-12	.002	4*	1000
-11	.002	4*	1000
-10	.003	4*	1000
-9	.004	4*	1000
-8	.006	4*	1000
-7	.009	6	111111
-6	.014	6	111110
-5	.020	5	11011
-4	.032	5	11010
-3	.054	4	1100
-2	.081	4	1011
-1	.121	3	010
0	.302	2	00
1	.119	3	011
2	.078	4	1010
3	.053	4	1001
4	.031	5	11101
5	.020	5	11100
6	.014	6	111101
7	.009	6	111100
8	.006	4*	1000
9	.004	4*	1000
10	.003	4*	1000
11	.002	4*	1000
12	.002	4*	1000
13	.001	4*	1000
14	.001	4*	1000
15	.001	4*	1000

```
  16      .000          4*      1000
  17      .000          4*      1000
  18      .000          4*      1000
  19      .000          4*      1000
  20      .000          4*      1000

 *GROUPED PROBABILITY =    .043  CODE LENGTH =   4
```

HUFFMAN CODES FOR SSOIAM

AVERAGE CODE LENGTH 3.046 ENTROPY 2.801

LEVEL	PROB.	LENGTH	
-20	.000	4*	1100
-19	.000	4*	1100
-18	.000	4*	1100
-17	.000	4*	1100
-16	.000	4*	1100
-15	.000	4*	1100
-14	.000	4*	1100
-13	.000	4*	1100
-12	.000	4*	1100
-11	.000	4*	1100
-10	.000	4*	1100
-9	.001	4*	1100
-8	.001	4*	1100
-7	.002	4*	1100
-6	.003	4*	1100
-5	.006	4*	1100
-4	.012	4*	1100
-3	.028	5	11111
-2	.069	4	1110
-1	.169	3	101
0	.316	2	00
1	.223	2	01
2	.096	3	100
3	.039	4	1101
4	.017	5	11110
5	.008	4*	1100
6	.004	4*	1100
7	.002	4*	1100
8	.001	4*	1100
9	.001	4*	1100
10	.000	4*	1100
11	.000	4*	1100
12	.000	4*	1100
13	.000	4*	1100
14	.000	4*	1100
15	.000	4*	1100

```
 16      .000         4*      1100
 17      .000         4*      1100
 18      .000         4*      1100
 19      .000         4*      1100
 20      .000         4*      1100

*GROUPED PROBABILITY =   .043  CODE LENGTH =  4
```

SSDI COMPRESSION STATISTICS

CASE ID: 'C[illegible]- 35X35 *SSDI*

OUTPUT BITS/PIXEL =3.0 BUF1 , =3.5 BUF2

SCAN LINE	BITS IN BUFS		HUF. (BITS/PIX)	ADAPT. HUF. (BITS/PIX)	RICE (BITS/PIX)
1	1360	0	3.327	3.352	3.422
2	3211	239	3.340	3.421	3.574
3	5065	431	3.371	3.473	3.575
4	6723	527	3.329	3.387	3.514
5	8536	728	3.356	3.372	3.563
6	10400	980	3.364	3.475	3.578
7	12099	1066	3.305	3.357	3.527
8	13817	1173	3.326	3.385	3.533
9	15353	1097	3.316	3.338	3.476
10	16947	979	3.283	3.283	3.453
11	18433	953	3.327	3.330	3.493
12	20235	1143	3.342	3.413	3.559
13	21708	1004	3.270	3.338	3.457
14	23205	889	3.274	3.371	3.464
15	24662	734	3.307	3.324	3.452
16	26335	795	3.336	3.366	3.519
17	27821	669	3.302	3.326	3.461
18	29649	884	3.390	3.440	3.567
19	31468	1093	3.352	3.375	3.565
20	33330	1342	3.366	3.400	3.578
21	35276	1676	3.339	3.403	3.604
22	37207	1993	3.339	3.414	3.599
23	39109	2285	3.356	3.429	3.590
24	40945	2509	3.340	3.384	3.569
25	42940	2893	3.371	3.402	3.619
26	44983	3303	3.398	3.449	3.634
27	47141	3669	3.433	3.489	3.669
28	48954	4070	3.396	3.442	3.562
29	50998	4403	3.383	3.415	3.603
30	52946	4728	3.410	3.435	3.604
31	54733	4993	3.374	3.363	3.579
32	56877	5545	3.474	3.514	3.671
33	58285	5341	3.436	3.517	3.437
34	59548	4992	3.400	3.504	3.392
35	61723	5555	3.404	3.484	3.675

36	64179	6279	3.440	3.532	3.756
37	65814	6422	3.438	3.577	3.513
38	67322	6318	3.467	3.563	3.468
39	68324	5708	3.449	3.512	3.311
40	70424	6196	3.398	3.498	3.651
41	72505	6655	3.382	3.467	3.645
42	74545	7054	3.417	3.472	3.633
43	77086	8022	3.454	3.565	3.788
44	79741	9065	3.479	3.589	3.824
45	81329	9051	3.473	3.561	3.496
46	83944	10044	3.480	3.559	3.808
47	86283	10771	3.446	3.540	3.725
48	88520	11396	3.440	3.439	3.654
49	90963	12227	3.471	3.521	3.758
50	93361	13013	3.454	3.556	3.744
51	95505	13545	3.414	3.490	3.665
52	97466	13894	3.395	3.470	3.608
53	99266	14082	3.313	3.366	3.558
54	101008	14212	3.354	3.291	3.540
55	102652	14244	3.299	3.345	3.510
56	104544	14524	3.356	3.404	3.587
57	106423	14791	3.330	3.392	3.583
58	108074	14770	3.342	3.393	3.493
59	109915	15059	3.363	3.377	3.590
60	111087	14619	3.422	3.559	3.364
61	112626	14546	3.439	3.509	3.477
62	113960	14268	3.444	3.528	3.414
63	115040	13726	3.375	3.437	3.335
64	116141	13225	3.354	3.428	3.342
65	117953	13425	3.337	3.422	3.562
66	120048	13509	3.382	3.469	3.650
67	122369	14617	3.422	3.484	3.720
68	123677	14313	3.404	3.501	3.406
69	125695	14719	3.377	3.441	3.626
70	128045	15457	3.459	3.548	3.729
71	129307	15107	3.441	3.499	3.391
72	130702	14890	3.475	3.503	3.433
73	133075	14631	3.419	3.477	3.420
74	134519	15483	3.457	3.555	3.764
75	136307	15559	3.480	3.640	3.524
76	137679	15419	3.470	3.570	3.457
77	138853	14391	3.386	3.492	3.364
78	140229	14745	3.453	3.536	3.427
79	141748	14652	3.399	3.479	3.471

80	143500	14792	3.494	3.637	3.543
81	145038	14718	3.430	3.592	3.477
82	146676	14744	3.454	3.582	3.508
83	148308	14764	3.476	3.618	3.506
84	150035	14979	3.498	3.601	3.536
85	151662	14894	3.460	3.570	3.505
86	153573	15193	3.525	3.666	3.593
87	155533	15541	3.529	3.669	3.608
88	157379	15775	3.536	3.708	3.573
89	159263	16047	3.535	3.692	3.584
90	161290	16463	3.550	3.668	3.629
91	163217	16777	3.505	3.672	3.598
92	165169	17117	3.548	3.716	3.605
93	166893	17229	3.523	3.670	3.535
94	168857	17581	3.546	3.694	3.609
95	170824	17936	3.542	3.696	3.610
96	172121	17621	3.543	3.671	3.402
97	174228	18116	3.493	3.674	3.654
98	176269	18549	3.508	3.702	3.633
99	177941	18605	3.461	3.631	3.519
100	179679	18731	3.487	3.590	3.539
101	181345	18785	3.483	3.576	3.517
102	182942	18770	3.480	3.584	3.495
103	184485	18701	3.457	3.533	3.479
104	186337	18941	3.488	3.632	3.574
105	188229	19221	3.438	3.651	3.597
106	190279	19659	3.541	3.713	3.636
107	192005	19774	3.516	3.693	3.576
108	193804	19960	3.449	3.567	3.558
109	196629	21173	3.518	3.672	3.876
110	198282	21214	3.475	3.597	3.513
111	200615	21935	3.437	3.552	3.724
112	202932	22640	3.422	3.544	3.719
113	205275	23371	3.425	3.567	3.727
114	207795	24380	3.444	3.551	3.782
115	209769	24221	3.415	3.544	3.482
116	211187	24447	3.479	3.608	3.570
117	213705	25553	3.470	3.632	3.843
118	215725	25761	3.503	3.640	3.565
119	216762	25186	3.464	3.594	3.322
120	218239	25151	3.448	3.590	3.489
121	219827	25037	3.380	3.516	3.465

122	220800	24339	3.421	3.587	3.299
123	221783	23764	3.441	3.598	3.306
124	222519	22893	3.396	3.510	3.227
125	223515	22257	3.408	3.563	3.309
126	224428	21568	3.391	3.513	3.283
127	225257	20765	3.344	3.484	3.257
128	227514	21430	3.433	3.526	3.700
129	229370	21674	3.354	3.431	3.576
130	231238	21930	3.346	3.347	3.579
131	232938	22018	3.329	3.359	3.527
132	234686	22154	3.355	3.375	3.542
133	236525	22381	3.319	3.291	3.570
134	238671	22915	3.356	3.380	3.666
135	239849	22481	3.357	3.394	3.365
136	241003	22023	3.378	3.414	3.358
137	242234	21642	3.360	3.400	3.382
138	243652	21478	3.428	3.588	3.449
139	245010	21194	3.394	3.515	3.412
140	245933	20505	3.312	3.317	3.286
141	246966	19926	3.348	3.340	3.320
142	247660	19009	3.271	3.214	3.215
143	249255	18991	3.302	3.258	3.495
144	251225	19349	3.315	3.325	3.611
145	252162	18674	3.292	3.293	3.291
146	252951	17851	3.279	3.280	3.245
147	254746	18034	3.316	3.312	3.557
148	256517	18193	3.328	3.299	3.549
149	258129	18193	3.300	3.249	3.500
150	259805	18257	3.330	3.388	3.520
151	261342	18182	3.296	3.260	3.477
152	262911	18139	3.246	3.167	3.487
153	264654	18370	3.301	3.301	3.541
154	265726	17730	3.358	3.349	3.333
155	267259	17651	3.265	3.216	3.475
156	269100	17880	3.338	3.354	3.571
157	270726	17894	3.295	3.229	3.504
158	272308	17864	3.290	3.262	3.491
159	273978	17982	3.337	3.315	3.505
160	275509	17841	3.308	3.214	3.487
161	277114	17834	3.322	3.305	3.498
162	278940	18048	3.350	3.272	3.566
163	280035	17531	3.333	3.316	3.340
164	281130	17014	3.379	3.435	3.340
165	282312	16584	3.357	3.439	3.367

166	282469	16123	3.358	3.402	3.259
167	284219	15267	3.272	3.337	3.333
168	285324	14760	3.330	3.413	3.343
169	286354	14178	3.307	3.353	3.319
170	287427	13639	3.372	3.415	3.333
171	288496	13094	3.346	3.366	3.332
172	290268	13256	3.328	3.335	3.550
173	292079	13455	3.316	3.354	3.562
174	293965	13729	3.322	3.367	3.555
175	295443	13595	3.231	3.208	3.458
176	297052	13592	3.311	3.303	3.499
177	298729	13656	3.295	3.288	3.570
178	300439	13755	3.335	3.213	3.531
179	302059	13763	3.300	3.264	3.502
180	303360	13952	3.321	3.257	3.559
181	305472	13952	3.257	3.269	3.500
182	307261	14129	3.299	3.309	3.535
183	309020	14276	3.255	3.287	3.546
184	310814	14458	3.290	3.259	3.556
185	312591	14623	3.321	3.318	3.551
186	313417	13837	3.252	3.227	3.256
187	314987	13795	3.278	3.196	3.457
188	316446	13642	3.245	3.187	3.453
189	318234	13818	3.291	3.291	3.555
190	320084	14056	3.324	3.301	3.574
191	321978	14339	3.367	3.383	3.587
192	323778	14526	3.306	3.315	3.558
193	325180	14316	3.255	3.159	3.435
194	326873	14395	3.333	3.370	3.525
195	328496	14409	3.304	3.268	3.504
196	330142	14442	3.333	3.269	3.511
197	331692	14330	3.281	3.278	3.481
198	333427	14503	3.315	3.315	3.538
199	335237	14801	3.325	3.297	3.597
200	337370	15072	3.317	3.328	3.584
201	339111	15351	3.308	3.350	3.587
202	341135	15763	3.365	3.407	3.628
203	342890	15906	3.301	3.290	3.544
204	344766	16170	3.342	3.390	3.582
205	346275	16168	3.267	3.200	3.499
206	348056	16236	3.371	3.280	3.521
207	349525	16502	3.367	3.356	3.583

208	351645	16601	3.326	3.335	3.530
209	353351	16695	3.306	3.334	3.529
210	355068	16800	3.315	3.361	3.533
211	356616	16736	3.298	3.308	3.480
212	358223	16731	3.292	3.279	3.498
213	359811	16707	3.279	3.285	3.493
214	361307	16591	3.251	3.241	3.464
215	362852	16524	3.259	3.257	3.479
216	364662	16722	3.335	3.361	3.561
217	366289	16737	3.287	3.290	3.505
218	368162	16998	3.330	3.374	3.581
219	370162	17386	3.373	3.446	3.620
220	371979	17591	3.346	3.387	3.564
221	373778	17778	3.335	3.408	3.558
222	375562	17950	3.371	3.395	3.553
223	377353	18129	3.364	3.398	3.556
224	379353	18517	3.393	3.446	3.620
225	381361	18913	3.372	3.436	3.623
226	383222	19162	3.342	3.408	3.577
227	385059	19387	3.334	3.369	3.570
228	386804	19520	3.295	3.347	3.541
229	388547	19951	3.362	3.438	3.634
230	391077	20569	3.415	3.489	3.692
231	393296	21176	3.404	3.489	3.688
232	394574	20842	3.405	3.496	3.396
233	396617	21273	3.385	3.466	3.634
234	397879	20923	3.377	3.461	3.391
235	399974	21306	3.340	3.408	3.619
236	401777	21597	3.371	3.439	3.590
237	403600	21808	3.335	3.398	3.565
238	405480	22076	3.362	3.402	3.583
239	407432	22416	3.390	3.445	3.605
240	409364	22736	3.363	3.399	3.599
241	411324	23084	3.306	3.379	3.608
242	413137	23295	3.300	3.370	3.562
243	414754	23290	3.281	3.337	3.502
244	416342	23266	3.316	3.330	3.493
245	417997	23309	3.299	3.282	3.482
246	419943	23643	3.398	3.504	3.635
247	421879	23967	3.356	3.402	3.600
248	423810	24286	3.358	3.403	3.599
249	425621	24485	3.335	3.380	3.562
250	427663	24915	3.394	3.440	3.633
251	429424	25064	3.355	3.392	3.546

252	431426	25454	3.376	3.474	3.621
253	433269	25685	3.337	3.391	3.572
254	435201	26005	3.373	3.419	3.599
255	437160	26352	3.387	3.440	3.609
256	438985	26565	3.356	3.432	3.566
257	440784	26752	3.352	3.380	3.558
258	442710	27066	3.347	3.426	3.597
259	444555	27299	3.336	3.380	3.572
260	446807	27939	3.416	3.515	3.699
261	448927	28457	3.391	3.487	3.661
262	450935	28843	3.367	3.441	3.620
263	452741	29037	3.337	3.389	3.560
264	454583	29267	3.340	3.385	3.571
265	456585	29657	3.365	3.399	3.621
266	458664	30124	3.385	3.444	3.645
267	460794	30643	3.387	3.473	3.661
268	462978	31211	3.436	3.497	3.676
269	465190	31914	3.388	3.464	3.687
270	467665	32677	3.464	3.525	3.758
271	469567	33067	3.374	3.434	3.621
272	471502	33290	3.352	3.416	3.569
273	473330	33506	3.386	3.446	3.567
274	475137	33701	3.355	3.377	3.560
275	476717	33669	3.335	3.359	3.490
276	478564	33904	3.377	3.399	3.573
277	480514	34242	3.358	3.395	3.605
278	482547	34667	3.402	3.444	3.631
279	484507	35095	3.402	3.449	3.634
280	486715	35607	3.426	3.478	3.659
281	488926	36305	3.449	3.506	3.686
282	491346	37014	3.470	3.518	3.751
283	493722	37778	3.431	3.515	3.737
284	496257	38701	3.467	3.564	3.786
285	498533	39365	3.454	3.548	3.705
286	500080	39300	3.471	3.525	3.480
287	502451	40059	3.437	3.512	3.735
288	504808	40904	3.487	3.545	3.731
289	506896	41280	3.379	3.446	3.648
290	509035	41807	3.417	3.453	3.653
291	511370	42528	3.459	3.529	3.727
292	513921	43479	3.493	3.577	3.792
293	516332	44268	3.466	3.553	3.749
294	518772	45097	3.462	3.525	3.757
295	521146	45659	3.450	3.533	3.736

296	522338	46598	3.511	3.589	3.850
297	524577	48065	3.519	3.680	3.834
298	526644	48520	3.506	3.613	3.641
299	531335	49599	3.549	3.695	3.835
300	533371	50023	3.550	3.655	3.632
301	535473	50513	3.548	3.667	3.652
302	537642	51070	3.572	3.696	3.673
303	540434	52230	3.538	3.638	3.860
304	543462	53666	3.547	3.657	3.945
305	546256	54848	3.535	3.646	3.867
306	549291	56271	3.572	3.678	3.941
307	552163	57531	3.546	3.663	3.891
308	555429	59185	3.587	3.731	4.013
309	558404	60548	3.544	3.686	3.923
310	561450	61982	3.582	3.735	3.945
311	564342	63262	3.574	3.694	3.897
312	567493	64761	3.592	3.672	3.965
313	570444	66140	3.583	3.702	3.928
314	573510	67594	3.621	3.743	3.951
315	575447	67919	3.560	3.659	3.601
316	577420	68230	3.582	3.663	3.612
317	579465	68713	3.593	3.637	3.634
318	582335	69971	3.560	3.635	3.890
319	585041	71045	3.518	3.599	3.839
320	587904	72316	3.569	3.668	3.888
321	590331	73131	3.487	3.559	3.753
322	592853	74051	3.523	3.598	3.785
323	595524	75100	3.548	3.606	3.825
324	598290	76254	3.537	3.617	3.858
325	601092	77444	3.550	3.657	3.869
326	603753	78493	3.557	3.635	3.825
327	606126	79654	3.533	3.648	3.860
328	609164	80680	3.549	3.638	3.918
329	611665	81569	3.514	3.577	3.776
330	614416	82709	3.550	3.638	3.853
331	617133	83813	3.553	3.629	3.843
332	618936	84004	3.522	3.618	3.559
333	620573	84129	3.521	3.585	3.537
334	623336	85240	3.579	3.627	3.845
335	626115	85347	3.547	3.653	3.843
336	628155	86775	3.587	3.677	3.633
337	629885	86993	3.494	3.600	3.537

338	631917	87212	3.616	3.669	3.630
339	633748	87532	3.538	3.629	3.568
340	635426	87598	3.519	3.583	3.520
341	636943	87503	3.497	3.551	3.471
342	639557	88205	3.479	3.524	3.718
343	641783	89119	3.496	3.545	3.783
344	644296	90020	3.503	3.566	3.779
345	646809	90921	3.494	3.570	3.779
346	649536	92036	3.518	3.585	3.846
347	651067	91955	3.463	3.543	3.475
348	652948	92224	3.569	3.656	3.583
349	654729	92393	3.503	3.596	3.553
350	656394	92446	3.527	3.610	3.516
351	657805	92245	3.467	3.522	3.438
352	659242	92170	3.489	3.525	3.477
353	660521	91737	3.425	3.471	3.366
354	662692	93296	3.447	3.491	3.672
355	665051	93043	3.470	3.509	3.722
356	666641	93021	3.474	3.565	3.493
357	668259	93027	3.518	3.629	3.502
358	669873	93029	3.505	3.558	3.501
359	671314	92858	3.453	3.527	3.447
360	673102	93034	3.578	3.622	3.555
361	674780	93100	3.514	3.593	3.520
362	676275	93083	3.527	3.595	3.495
363	677728	92624	3.469	3.516	3.420
364	679114	92598	3.465	3.482	3.430
365	680482	92354	3.503	3.520	3.424
366	682048	92308	3.509	3.523	3.486
367	683209	91657	3.395	3.422	3.360
368	685358	92294	3.444	3.493	3.467
369	686541	91965	3.405	3.456	3.367
370	688476	92288	3.410	3.443	3.600
371	690162	92262	3.324	3.346	3.523
372	692144	92732	3.456	3.492	3.615
373	692952	91928	3.342	3.360	3.251
374	693909	91273	3.271	3.419	3.257
375	694907	90659	3.358	3.424	3.310
376	695912	90052	3.386	3.441	3.312
377	696965	89433	3.381	3.449	3.224
378	698081	88997	3.393	3.437	3.349
379	699191	88495	3.400	3.473	3.344
380	700265	87957	3.390	3.429	3.333
381	701347	87427	3.410	3.440	3.336

382	702409	86877	3.380	3.398	3.329
383	703439	86295	3.337	3.405	3.319
384	704540	85784	3.425	3.470	3.342
385	705681	85313	3.393	3.421	3.354
386	707071	85091	3.467	3.497	3.421
387	708249	84657	3.401	3.454	3.355
388	709554	84350	3.402	3.483	3.405
389	710713	83597	3.397	3.461	3.359
390	712000	83572	3.455	3.489	3.399
391	713271	83331	3.473	3.451	3.425
392	714314	83262	3.471	3.564	3.479
393	716247	82983	3.424	3.501	3.413
394	717598	82722	3.456	3.520	3.419
395	719023	82535	3.445	3.513	3.442
396	720358	82268	3.455	3.524	3.417
397	721674	81962	3.429	3.471	3.405
398	723006	81682	3.414	3.480	3.413
399	724394	81358	3.411	3.478	3.400
400	725642	81094	3.445	3.523	3.418
401	726966	80806	3.413	3.485	3.411
402	728344	80572	3.494	3.552	3.427
403	729453	80069	3.374	3.392	3.344
404	731747	80751	3.440	3.484	3.712
405	732778	80170	3.342	3.400	3.320
406	734688	80468	3.388	3.440	3.592
407	736672	80840	3.381	3.404	3.615
408	738935	81491	3.459	3.511	3.702
409	741342	82286	3.450	3.494	3.747
410	742572	81904	3.475	3.501	3.382
411	744678	82398	3.430	3.461	3.653
412	746960	83068	3.456	3.514	3.708
413	748807	83203	3.395	3.435	3.573
414	750958	83842	3.444	3.472	3.667
415	753158	84420	3.435	3.453	3.682
416	755234	84894	3.436	3.462	3.644
417	757217	85245	3.391	3.390	3.615
418	759461	85897	3.429	3.481	3.696
419	760791	85615	3.352	3.405	3.413
420	762926	86128	3.443	3.462	3.662
421	765222	86823	3.437	3.470	3.712
422	767646	87533	3.493	3.542	3.720
423	769776	88152	3.489	3.502	3.692

424	772174	88647	3.480	3.542	3.745
425	774415	89557	3.443	3.503	3.694
426	776880	90420	3.522	3.565	3.765
427	779279	91167	3.491	3.495	3.722
428	781753	92069	3.519	3.560	3.720
429	784005	92709	3.471	3.517	3.699
430	786107	93194	3.473	3.540	3.650
431	788214	93694	3.449	3.466	3.655
432	790403	94273	3.449	3.488	3.679
433	792622	94878	3.448	3.472	3.698
434	794914	95539	3.470	3.513	3.711
435	796995	.96027	3.444.	3.469	3.649
436	799030	96450	3.429	3.476	3.621
437	801084	96897	3.432	3.442	3.637
·438	803322	97518	3.510	3.531	3.694
439	805613	98197	3.477	3.488	3.711
440	807932	98804	3.460	3.479	3.688
441	810027	99337	3.488	3.484	3.681
442	812032	99780	3.438	3.450	3.622
443	813646	99782	3.342	3.312	3.501
444	815637	100161	3.430	3.409	3.618
445	817727	100649	3.433	3.413	3.651
446	819664	100964	3.428	3.418	3.598
447	821754	101442	3.447	3.416	3.649
448	823566	101642	3.377	3.393	3.562
449	825647	102106	3.399	3.435	3.644
450	827971	102823	3.501	3.522	3.737
451	830021	103261	3.474	3.432	3.636
452	832227	103965	3.465	3.467	3.687
453	834387	104403	3.446	3.461	3.667
454	836500	104904	3.407	3.472	3.655
455	838600	105392	3.430	3.445	3.651
456	840990	106070	3.462	3.438	3.710
457	843067	106635	3.487	3.494	3.475
458	845248	107304	3.439	3.505	3.709
459	846717	107061	3.477	3.487	3.425
460	849001	107723	3.469	3.505	3.703
461	851223	108413	3.489	3.519	3.711
462	853407	108910	3.442	3.466	3.654
463	855575	109371	3.534	3.531	3.767
464	857674	110558	3.539	3.556	3.744
465	860440	111112	3.446	3.466	3.672
466	862670	111730	3.474	3.544	3.692
467	864785	112232	3.457	3.467	3.656

468	867045	112881	3.501	3.504	3.701
469	869239	113563	3.512	3.502	3.712
470	871466	114078	3.447	3.459	3.660
471	873636	114636	3.450	3.477	3.673
472	875737	115125	3.415	3.446	3.652
473	877928	115704	3.461	3.489	3.680
474	880354	116518	3.511	3.563	3.752
475	882542	117094	3.447	3.489	3.679
476	884765	117705	3.456	3.503	3.690
477	887138	118466	3.475	3.496	3.736
478	889714	119430	3.540	3.620	3.799
479	891912	120016	3.457	3.500	3.682
480	894176	120663	3.445	3.487	3.702
481	896407	121287	3.465	3.501	3.692
482	898801	122069	3.497	3.524	3.743
483	901023	122679	3.473	3.495	3.689
484	903307	123351	3.503	3.556	3.708
485	905632	124064	3.521	3.539	3.721
486	908090	124910	3.533	3.525	3.762
487	910329	125537	3.488	3.507	3.694
488	912552	126148	3.494	3.514	3.690
489	914596	126580	3.440	3.450	3.634
490	916625	126997	3.435	3.495	3.629
491	918654	127414	3.461	3.480	3.629
492	920719	127867	3.444	3.444	3.641
493	922867	128403	3.450	3.458	3.666
494	925456	129380	3.559	3.622	3.803
495	927529	129841	3.484	3.484	3.643
496	929691	130693	3.516	3.561	3.764
497	932436	131524	3.526	3.571	3.758
498	934809	132284	3.484	3.545	3.736
499	937157	133021	3.490	3.509	3.729
500	939912	134164	3.596	3.636	3.855
501	942407	135047	3.547	3.587	3.774
502	944631	135859	3.525	3.605	3.752
503	947160	136576	3.505	3.553	3.722
504	949686	137430	3.544	3.571	3.782
505	952039	138281	3.505	3.547	3.745
506	954358	139133	3.550	3.597	3.766
507	956953	139921	3.542	3.573	3.743
508	959404	140760	3.530	3.595	3.760
509	961656	141400	3.510	3.544	3.699

510	964274	142346	3.554	3.595	3.752
511	966660	143130	3.554	3.584	3.759
512	969334	144242	3.602	3.646	3.829
513	971934	145230	3.570	3.587	3.806
514	974316	146000	3.526	3.609	3.739
515	976683	146756	3.503	3.535	3.734
516	979138	147598	3.509	3.561	3.761
517	981635	148482	3.544	3.572	3.775
518	984165	149401	3.551	3.588	3.785
519	986654	150278	3.533	3.572	3.772
520	989018	151030	3.578	3.601	3.733
521	991097	151497	3.437.	3.467	3.645
522	993393	152181	3.524	3.507	3.732
523	995761	152937	3.487	3.506	3.734
524	998240	153904	3.526	3.573	3.769
525	1000681	154623	3.471	3.578	3.757
526	1003297	155637	3.530	3.631	3.811
527	1004674	155402	3.439	3.502	3.437.
528	1006975	156091	3.479	3.528	3.714
529	1009543	157047	3.522	3.584	3.797
530	1011936	157878	3.493	3.572	3.742
531	1014178	158458	3.440	3.493	3.695
532	1016439	159107	3.459	3.494	3.701
533	1018719	159775	3.447	3.537	3.707
534	1021254	160698	3.518	3.592	3.786
535	1023592	161474	3.486	3.552	3.725
536	1025794	162014	3.444	3.496	3.683
537	1028122	162730	3.498	3.508	3.722
538	1030245	163241	3.431	3.494	3.658
539	1032232	163606	3.400	3.431	3.613
540	1034256	164028	3.437	3.453	3.531
541	1036198	164248	3.403	3.407	3.599
542	1038150	164698	3.414	3.428	3.609
543	1040038	164974	3.376	3.396	3.586
544	1042085	165409	3.355	3.474	3.635
545	1044157	165869	3.395	3.464	3.643
546	1046189	166330	3.369	3.408	3.630
547	1048219	166707	3.344	3.375	3.630
548	1050299	167275	3.393	3.463	3.676
549	1052341	167605	3.364	3.427	3.602
550	1054252	168004	3.373	3.456	3.624
551	1056173	168213	3.342	3.409	3.565
552	1057981	168209	3.341	3.370	3.530
553	1059432	168243	3.330	3.326	3.481

554	1061047	168951	3.370	3.320	3.507
555	1062807	168394	3.352	3.330	3.544
556	1064619	168599	3.350	3.404	3.564
557	1066179	168547	3.293	3.330	3.484
558	1067485	168446	3.315	3.355	3.468
559	1069284	168478	3.317	3.325	3.495
560	1070858	168390	3.325	3.333	3.488
561	1072375	168235	3.304	3.296	3.471
562	1073743	168051	3.247	3.279	3.424
563	1074990	167684	3.248	3.325	3.387
564	1076482	167566	3.272	3.297	3.463
565	1077642	167314	3.330	3.225	3.422
566	1079301	167251	3.292	3.209	3.480
567	1080587	166835	3.231	3.238	3.371
568	1081672	166308	3.229	3.246	3.337
569	1082680	165704	3.210	3.189	3.313
570	1083982	165394	3.247	3.264	3.404
571	1085389	165189	3.284	3.280	3.436
572	1086767	164955	3.271	3.266	3.427
573	1088290	164855	3.289	3.272	3.469
574	1089744	164708	3.291	3.291	3.454
575	1090805	164157	3.223	3.166	3.329
576	1091912	163653	3.234	3.154	3.344
577	1092774	162903	3.154	3.036	3.267
578	1094086	162602	3.266	3.141	3.407
579	1095280	162184	3.242	3.170	3.370

```
OVERALL DATA COMPRESSION

   ADAPTIVE HUFFMAN =  3.47  RICE =  3.59

PERCENT OCCURRENCE OF RICE MODES

  FS    FSC   FSCB SP61 SP43 SP34
 10.4 83.4  0.0  6.2  0.0  0.0
```

APPENDIX C
ALGORITHM FOR UNPACKING AND DECODING

The algorithm shown in Figure C-1 unpacks and decodes the packed, encoded data samples. Reading and writing of data blocks is double buffered and overlapped with processing. A table look-up operation is performed to decode compressed picture information and obtain an index for unpacking. The compressed picture information is further processed to reconstruct each picture.

The algorithm details the time critical operations of unpacking and decoding. The following discussion explains how unpacking and decoding can be performed on a minicomputer to minimize time and cost. For purposes of discussion, data is read from one of two blocks (j pointer) for processing and written to one of two blocks (k pointer). The four data blocks reside in the main memory to buffer information between the processor and input/output devices.

Eight registers contain the immediate information required for processing successive data samples. The following table describes the the role of each register. The word length of the machine is denoted by n.

R1 accumulator.

R2 s, shift index for unpacking.

R3 a } a register pair to contain from n to 2n
R4 b } bits of packed, encoded data samples. The high order bit can be shifted left out of b into the low order bit position in a.

R5 d, the number of left-justified data bits remaining in b.

R6 i, the index into block j for moving data to register b.

R7 j, input data block pointer.

R8 k, output data block pointer.

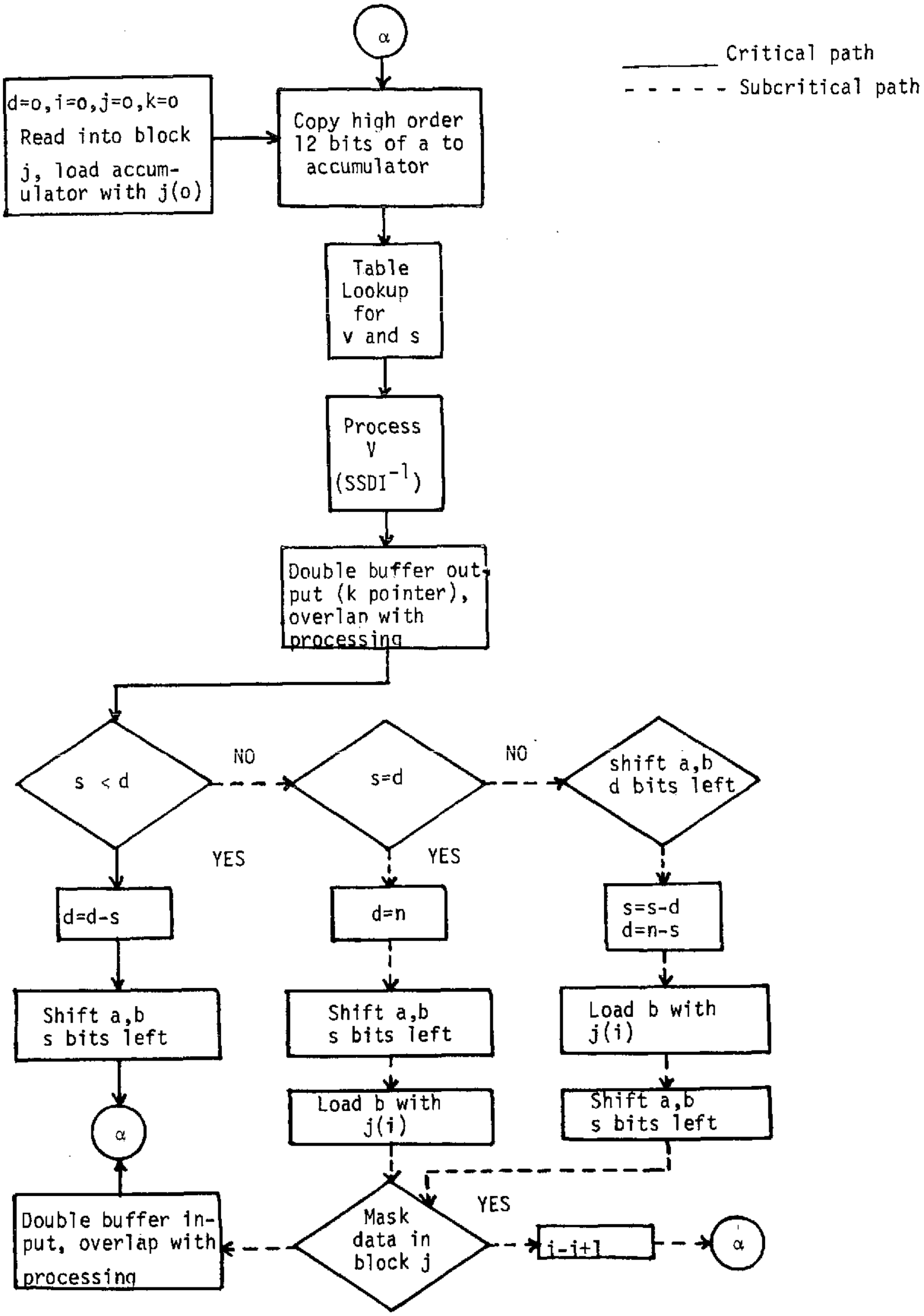


Figure C-1. Reconstruction Algorithm Flow

At START, data is read into the input buffer, pointers are initialized, and the first word is copied into a from the buffer. The N=12 high order bits of a are used as an index to a table to obtain the data value (v) and length (s) of the first code word. The value v is used in picture reconstruction according to the inverse SSDI algorithm as previously discussed. The length of the code word, s, is used in the following sequence of steps to unpack the next code word. The parameter s represents the number of bits that the register pair a-b must be shifted left. Before such shifting, however, it is necessary to determine when another word of packed, encoded data samples must be loaded into the b register. This question is resolved by comparing s and d (the number of data bits in b). In the general case s is less than d, so d is decremented, shifting is performed, and the algorithm returns to α to repeat the table look-up for the 12 high-order bits of a.

If s equals d, a is reset to the machine word length, a-b is shifted left s bits, and the next word from the input buffer is loaded into b. Before returning to α, the index i is incremented to point to the next word in the input buffer or another block is read if the current input buffer is empty.

If s is greater than d (initially d = 0) a-b is shifted until b is empty, b is loaded with another word, and a-b is shifted left the remaining s-d bits. The index i is incremented or another block is read before returning to α.

The critical path in this algorithm is the case where s is less than d, because the average code word length is less than one-third the machine word length. This path is shorter than the other two because no indexing or reading is required to service the b register. The next most frequently used path corresponds to the case of s greater than d. In all cases the alignment of code words in a is expedited by using the registers for immediate processing instead of accessing memory.

APPENDIX D
EFFICIENT GROUND-BASED DATA COMPRESSION OF IMAGES

The volume of data gathered by earth observation satellites such as the Earth Resources Technology Satellites (ERTS) produce major data handling problems both in ground data archiving and in the transmission of the data to the various investigators. Since each ground scene (100 nmi x 100 nmi) occupies four computer tapes, tape cost and storage facilities for the data produced became significant. The use of ground based data compression can diminish the magnitude of these problems.

To be acceptable for such processing, several requirements must be met by the compression algorithm. Of primary importance, the data compression technique must permit reconstruction of the data with archival fidelity, i.e., no error can be introduced into the data. The strictly information preserving SSDI and SSDIA algorithms meet this requirement.

A second constraint on the processing technique requires a moderately simple algorithm for compressing the ERTS data received on ground and a very efficient reconstruction algorithm. The processing time required for the initial compression which is performed only once can be substantially longer than the reconstruction time since the reconstruction of a given scene may be performed many times by different investigators.

The reconstruction will also be performed by different user processing facilities ranging from modest minicomputers to large computer systems. The algorithm should be applicable to virtually any computer, setting limits on storage requirements, word size, and the instruction set used.

A final requirement is that the compression achieved be sufficient to justify the added processing time required for the reconstruction. A compression of 2:1 would permit a saving of three magnetic tapes per scene.

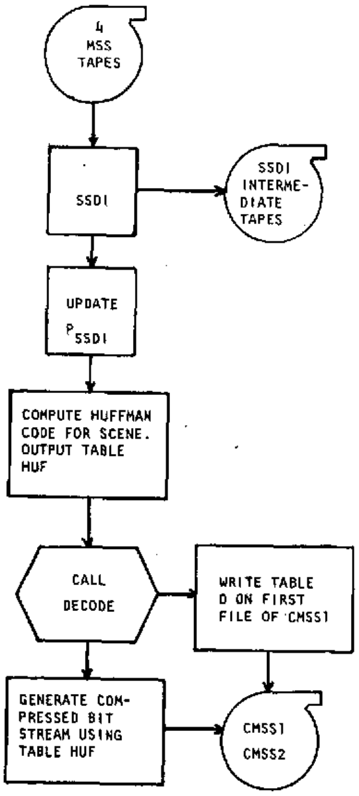
The SSDI transform combined with Huffmann encoding form the basis of a ground-based data compression system which can fulfill the above requirements. The SSDI is a very efficient algorithm, requiring only 7/4 subtracts/sample on the average and the inverse SSDI requires the same number of adds for reconstruction. This rapid encoding and reconstruction combined with a table look-up Huffman encoding and decoding of the data yield an algorithm which requires a minimal number of machine operations for processing the data while giving an information preserving data compression comparable to that achieved by more sophisticated algorithms. Huffman coding can be performed based on the statistics of the entire scene or adaptively, based on the statistics of blocks of data. Although locally adaptive Huffman coding gives a higher compression than coding based on global statistics of the entire scene, adaptive coding entails an increased complexity when encoding and decoding the data. Global Huffman coding is quite attractive for compression of ERTS data since the entire decoding operation is based on a single code.

The SSDIA algorithm permits a slightly higher degree of compression at the expense of increased storage and processing time. To perform the averaging operation, an entire scan line must be stored in each spectral band and an averaging operation performed for each input intensity. This additional load is negligible for the large computer used for encoding the data but could be a burden for reconstruction, especially if a small minicomputer is used with limited storage capabilities.

A second problem arises if SSDIA is used. Since the reconstruction algorithm of the SSDIA data presupposes information regarding the intensities of the pixels in the previous scan line, decoding must proceed from the first scan line of the image. If a user wanted to reconstruct only a portion of the data for his investigation, the SSDIA forces him to initially reconstruct all four tapes before he can select the area of interest. SSDI permits the operator to start at any scan line he desires and reconstruct only the segment of the tape he requires.

D.1 SSDI/HUFFMAN ENCODING OF MSS TAPES

Generation of the Huffman coded tapes from the MSS tapes would be performed once for each ERTS image received. The encoding operation should be efficiently performed with a minimum of operator interaction and the encoded data must be in a form that allows rapid decoding at user facilities. A technique will be described that meets these requirements while halving the number of tapes required for archiving and transmittal to the users of the data. Figure D-1 presents the basic data flow required for encoding the MSS tapes. The original four MSS tapes describing the scene are loaded onto the computer which subsequently generates the sequence of SSDI symbols. These symbols and auxiliary overhead data are written on intermediate tapes. Concurrent with the generation of the SSDI symbols, the probability of occurrence of each symbol is computed and stored in an array PSSDI. This array is of length 256 to include all possible SSDI levels which might occur.



4 MSS TAPES
SSDI
SSDI INTERME-DIATE TAPES
UPDATE P_{SSDI}
COMPUTE HUFFMAN CODE FOR SCENE. OUTPUT TABLE HUF
CALL DECODE
WRITE TABLE D ON FIRST FILE OF CMSS1
GENERATE COM-PRESSED BIT STREAM USING TABLE HUF
CMSS1
CMSS2

Figure D-1. Flow Chart for Encoding Data

After the input tapes have been read and the intermediate tapes generated, the distribution of SSDI symbols for the entire scene is used to generate the Huffman code to be used. The subroutine CODE generates the Huffman code desired and generates a table HUF of length 256. Each entry of table HUF contains the Huffman code associated with a particular SSDI symbol. Subroutine DECODE generates a table D which will be used by the decoding algorithm to enable table-look-up reconstruction of the data at the users processing facility.

The decoding table D is initially written as the first file on the output tape CMSS1. Following this, the Huffman encoded bit stream is written on the remainder of CMSS1 and CMSS2. The intermediate SSDI tapes and the table HUF are used concurrently to generate this compressed data. With each SSDI symbol read from the intermediate tape, an entry in array HUF is called where the entry is called with an argument corresponding to the SSDI symbol read. This array call returns the binary Huffman code associated with the symbol and that code word is written onto CMSS1 or CMSS2. The code words are allowed to overlap computer words in order to obtain a maximum compaction of the data.

To simplify reconstruction, several constraints are imposed on the code words assigned to SSDI symbols. First, no code word is allowed to have more than N bits. The choice of N will be discussed later. To permit this, low probability symbols are assigned a Huffman coded prefix C_L of at most N-8 bits. This prefix is then followed by true eight bits to allow separation of the symbols lumped under this prefix.* An efficient version of the Huffman coding algorithm has been developed and validated at TRW, as described in section 2.2.

*This technique preserves the instantaneous property of the code.

At the beginning of a scan line, the first picture element in each spectral band is transmitted as a direct 7 bit value. Occurrence of the beginning of a scan line is designated in the coded bit stream by C_L followed by eight ones. The next twenty-eight bits are the intensities of the first elements. All other picture elements in the scan line are encoded by the SSDI.

The table generated by subroutine DECODE contains an array of 2^N entries. The address of each entry corresponds to a sequence of N bits and the entry itself contains two quantities of information required by the reconstruction algorithm. One quantity is the first decodable symbol in that string of bits. Since the number of bits in an allowable Huffman code word can vary from one to N bits, each table entry is guaranteed to contain a decoded symbol. Although on the average several decodable symbols are contained in N bits, it appears that one decode per table entry permits a more economical reconstruction algorithm. The second quantity contained in a table entry is the number of shifts required to position the next decodable word so that it will begin in the first bit position of the next N bit string.

As an example, suppose that the first decodable word in the N bit sequence was C_K of length K bits. The table entry addressed by these N bits would return SSDI symbol S_K, corresponding to codeword C_K, and the number of shifts, K, required to position the next codeword as the header of the subsequent N bit sequence.*

The SSDI algorithm is very fast requiring seven subtract operations per set of four input intensity samples. The time required for these operations and the associated fetches from local storage is less than the time required for reading in the four input MSS tapes, implying that this stage of processing is limited by tape speed. The required calls to subroutines CODE and

*This encodinq operation takes into account the operations which must be performed by the decoding algorithm.

DECODE required a total time of about three seconds. Generation of the Huffaan coded bit stream using the table look up technique is limited by the time required to read in the intermediate SSDI tapes and write tapes CMSS1 and CMSS2. Thus, the entire encoding operation is essentially the same as the tape handling time.

The storage required for encoding tapes is minimal and is easily accommodated by any large computer. The generation of the SSDI symbols and the intermediate tapes requires the storage of eight input samples and four differences. The probability array PSSDI is of length 256. Array HUF is of length 256. Array D is of length 2^N, where N may typically equal 12. Each entry of D contains 8 bits giving the symbol decoded, followed by 4 bits giving the number of shifts required. The Huffman coding routine requires an additional array storage of length 1024.

D.2 DECODING AND RECONSTRUCTION PROCEDURE

The primary requirements imposed on the decoding algorithm are:

- Rapid reconstruction of the digital data
- Required storage within the limitations of modest minicomputer systems
- All computations and arrays limited to the single precision word length of the machines used
- Minimal operator interaction required.

These constraints will ensure that the algorithm selected can be implemented at all user installations. The goal of the algorithm development is a reconstruction technique that takes about the same time for reconstructing the compressed tapes as required for reading the original MSS tapes and which would require negligible additional effort by the computer operator.

Since there are over 27 million intensity samples per 100 nmi x 100 nmi scene, a very fast algorithm is necessary to minimize the number of machine operations required to decode each sample. Knowledge of the computer structure is essential so that an optimum flow can be established and an efficient division made between input/output operations and central processing.

The time of critical machine operations that are performed often must be minimized, perhaps at the expense of lengthening operations that are performed infrequently. One example of this involves the minimization of normal bit decoding at the expense of increasing the time for restoring pixel intensities at the beginning of a scan line since the latter procedure occurs infrequently.

A block diagram of the essential reconstruction flow is given in Figure D-2. A more detailed discussion of the machine operations involved in decoding is given in Appendix C.

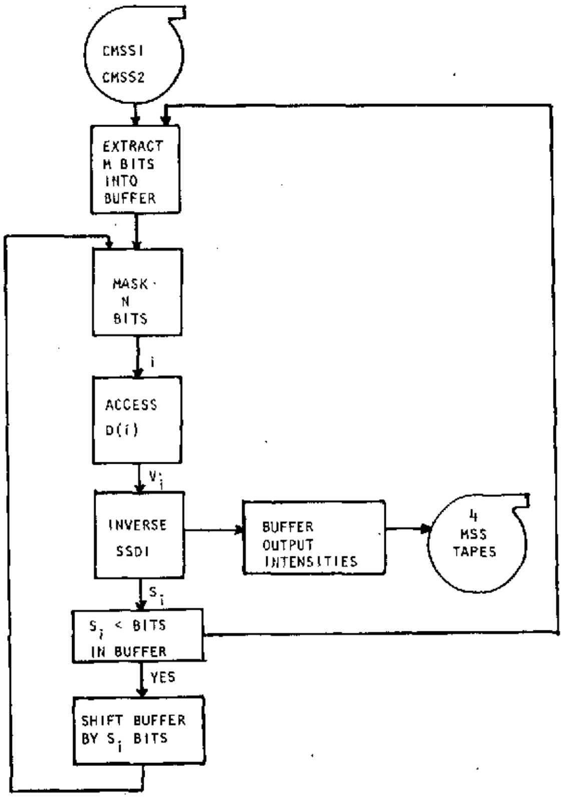


Figure D-2. Flow of the Decoding Algorithm

The input tapes contain the encoded bit stream in which no Huffman codeword contains more than N bits. A sequence of M > N bits is extracted at a time from the input tapes and stored in an input register. The value of M depends on the word length of the computer used and the number of registers available. Parameter M should be as large as possible in order to minimize the number of calls to the input tape which consume valuable processing time.

A sequence of N bits is extracted from the input register by appropriate masking operations. Decoding Table D, which has been read from the first file on the input tapes and put into local storage, is used to determine the first symbol contained in the N bit sequence. The N bit sequence is used as the address to access the proper table entry. This entry returns 12 bits of decoding information. The first 8 bits represent the first symbol V_i in the sequence. The next 4 bits represent the number of shifts S_i required to position the next N bit sequence for decoding. If symbol V_i has been coded with L_i bits, then $S_i = L_i \leq N$.

Symbol V_i is used to reconstruct the next intensity by the inverse SSDI algorithm. Registers can be used to store the four previously decoded intensities $\underline{I}$. The four SSDI differentials, packed consecutively on the input tapes, are added in the appropriate sequence to the previously reconstructed intensities to form the current intensities which then replace the previous values in the registers. The inverse SSDI reconstruction takes seven additional operations per set of four intensities. The reconstructed intensities are placed in a buffer for subsequent writing onto the four reconstructed MSS tapes.

In preparation for the next decoding operation, S_i bits are shifted out of the input buffer into the N bit register used to address Table D. The number of shifts S_i must be compared to the number of bits left in the input buffer to determine if the buffer contains at least S_i bits. If not, more data must be read into the buffer from the input tapes before the shifting occurs.

The normal algorithm flow, described above, occurs for all intensity values except for the set of four intensities at the beginning of a new scan line. A unique Huffman coded prefix is generated by the encoder at the beginning of each new scan line, and this prefix is followed by 28 bits which are the first four 7-bit intensity values in the scan line. Detection of this scan line prefix code by the decoding algorithm loads the following four intensities into the appropriate I registers, replacing the previous intensities stored there for the last four intensities in the preceding scan line.

D.3 SUMMARY

The encoding and decoding technique presented forms a practical and efficient method for the ground-based data compression of multispectral data with archival fidelity. The simplicity of the SSDI algorithm and the rapid decoding permitted by a Huffman table look-up, form the basis for an algorithm which can be rapidly performed.

The determination of the optimum block size N of code words is an important consideration. As N increases, compression increases, but Table D doubles in length each time N increases by one. Parameter N should be that value which allows an average data compression of at least 2:1 while permitting an array size D that can be accommodated by modest minicomputer systems. A value of N = 12 is currently being used.

Further simulation and optimization of the reconstruction algorithm will permit a determination of the minimal set of computations required. A study of available minicomputers would permit refining the set of machine instructions used, the number of registers allotted for local storage, and the allowable storage for Table D. In addition, the time required for reconstruction can be estimated for several typical computers.

While this discussion has concentrated on the use of the SSDI algorithm with Huffman coding based on the statistics of the entire scene, further study should be performed on the SSDIA algorithm and adaptive Huffman coding for ground reconstruction. These latter techniques permit an increased compression, but require a more complex reconstruction algorithm implying a longer reconstruction time. In addition, the SSDIA requires the storage of an entire scan line in each spectral band. The compression of four MSS tapes into one compressed tape provides both economic and archival benefits, assuming the use of efficient compression techniques applicable to a wide variety of computers. While further investigation and simulation is required, the proposed technique seems to represent a viable candidate for the ground compression of MSS digital tapes with archival fidelity.